Global Illumination by Bayesian Monte Carlo and Photon-Driven Irradiance Cache

Jonathan Brouillat

N° d'ordre : 3989 **ANNÉE 2009**

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Jonathan Brouillat

préparée à l'unité de recherche 6074 IRISA
Institut de recherche en informatique et systèmes aléatoires
IFSIC

Illumination Globale par Monte Carlo Bayésien et cache d'éclairement généré à partir d'une carte de photons

Thèse soutenue à Rennes le 24 novembre 2009

devant le jury composé de :

Charles HANSEN
Professeur – University of Utah / *rapporteur*

Nicolas HOLZSCHUCH
Chercheur – INRIA Rhône-Alpes / *rapporteur*

Bruno ARNALDI
Professeur – INSA de Rennes / *examinateur*

Christian BOUVILLE
Ingénieur – France Telecom Rennes / *examinateur*

Pascal GAUTRON
Ingénieur – Thomson Rennes / *examinateur*

Kadi BOUATOUCH
Professeur – Université de Rennes 1 / *directeur de thèse*

Contents

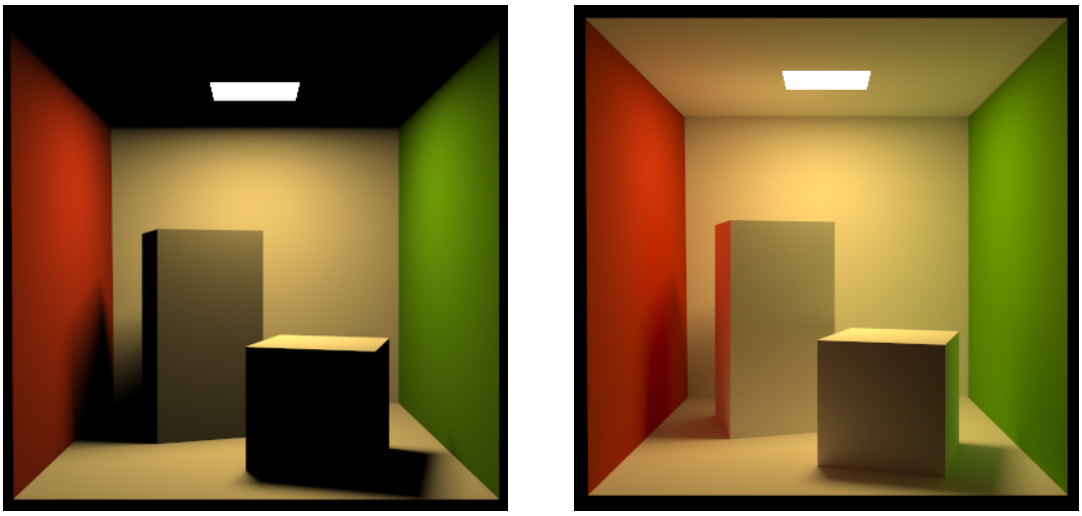
Chapter 1

Introduction

The goal of realistic rendering is to create images that are indistinguishable from reality. Many domains are particularly interested in realistic rendering, such as architectural design, cinema and video games. However, rendering realistic images requires an accurate representation and simulation of the interactions between light and matter. The aim is to simulate the multiples bounces of light in a scene. This process is called *global illumination*. In this thesis, our main interest is the fast and accurate computation of global illumination in complex scenes. More precisely, we investigate two approaches. First we propose to combine two widely used methods in order to exploit their advantages. Second, we investigate a new approach to variance reduction in Monte Carlo based rendering algorithms.

1.1 Global illumination

Global illumination has been an intensive research area over the last decades. The goal of this computation is to simulate how light is scattered and reflected off objects made up of various materials. The computation of global illumination can be divided into two parts: direct lighting and indirect lighting. Direct lighting considers only one bounce of light between the light source and the observer (Figure 1.1(a)). Consequently, many aspects of the lighting cannot be rendered by direct lighting only. Indirect lighting accounts for the multiple bounces of light in the scene (Figure 1.1(b)). This allows for the computation of complex lighting effects, such as indirect light sources, color bleeding and caustics. However, the computation of indirect lighting is very complex in the general case and requires several hours even on high performance computers. While, in the context the context of interactive applications such as video games, approximations are required for fast rendering. The work described in this thesis considers these two aspects. First we propose a method to compute a view independent global illumination solution covering the whole scene. This solution is then interactively displayed at runtime. Second, we investigate a new approach, called Bayesian Monte Carlo, to reduce the time needed for high quality rendering. It is the first time Bayesian Monte Carlo is applied to global illumination computation.



(a) Direct lighting only (b) With indirect lighting

Figure 1.1: Global illumination allows for the computation of complex lighting effects, such as indirect light sources, color bleeding and caustics. Images: Thiago Ize

1.2 Photon-Driven Irradiance Cache

Photon mapping and irradiance cache are two of the most used techniques in global illumination. We propose to exploit the advantages of both methods by combining them in a clever way. Indeed, *Photon mapping* [Jen96] allows to simulate any complex lighting by tracing particles through the scene. The irradiance incoming at a visible point is estimated by computing the density of the particles at a given location. Note that the lighting simulation pass is view-independent. The photon mapping technique provides multiple-bounce global illumination. However, the direct use of the information contained in the photon map yields noisy pictures. To obtain a high quality rendering, a costly pass of final gathering must be performed by tracing rays from each visible point. As for *Irradiance caching* [WRC88] it exploits the spatial coherence of the indirect lighting. More precisely, the irradiance is sparsely sampled at some points in the scene and stored into irradiance records, then interpolated for every other point. For each viewpoint a set of new records have to be computed which limits interactivity.

Our algorithm, Photon-Driven Irradiance Cache, builds a view-independent irradiance cache from a photon map. It allows fast and view independent preview of large photon maps. In addition, the generated cache requires almost no new computation during the rendering step. The cache can be rendered in real time, allowing for interactive walkthrough in static scenes.

1.3 Bayesian Monte Carlo for global illumination

Monte Carlo algorithms exhibit noise in the rendered pictures. This is a consequence of the inherent variance associated with the Monte Carlo estimators. Several techniques have been proposed to reduce the variance of these estimators. We investigate a new approach to variance

reduction based on Bayesian Monte Carlo.

Most variance reduction techniques consider only the values of the drawn samples (radiance along a ray), but not their relative positions. In addition, some of them can provide a different estimate with the same set of samples, which violates the Likehood Principle. Our variance reduction approach relies on Bayesian Monte Carlo (BMC) which avoids the above problems by exploiting a prior probabilistic knowledge on the incoming radiance at every point. Bayesian Monte Carlo accounts for the relative positions of the samples to reduce the variance of the final computation by considering the covariance properties of the radiance function. We show that the Bayesian Monte Carlo can improve global illumination results in terms of rendering quality compared to classical Monte Carlo. Finally we propose some extensions to other global illumination problems.

1.4 Thesis overview

This thesis is divided into 6 chapters. After this introduction, Chapter 2 presents a brief description of the main notions used in global illumination computation. Chapter 3 briefly recalls the most significant previous works in the field of global illumination, and details the methods of Photon Mapping and Irradiance Caching we use in our algorithm. We then present our first contribution, the Photon-Driven Irradiance Cache algorithm in Chapter 4. The Chapter 5 describes a new approach to variance reduction in global illumination by using Bayesian Monte Carlo, and proposes several extensions. Finally, Chapter 7 concludes this thesis and summarizes our contributions and directions for future work.

Chapter 2

Background

In this chapter, we describe the main notions used in global illumination computation. We also give the notations used through the different chapters. We start by giving some notions about radiometry and material properties, then describe the equation used for lighting simulation.

In a given environment, light propagates, starting from light sources and bouncing on objects present in the scene. The description of this distribution of radiance makes possible rendering of 3D scenes viewed from an arbitrary camera. Rendering consists in evaluating the radiance along camera rays, starting from the camera position and going through the camera view plane. First we give some basics about radiometry that are used to introduce the light transport equation. We then briefly present the notion of Bidirectional Reflectance Distribution Function (BRDF). This leads us to the light transport equation and some light source models used in different situations. Finally we introduce the notion of Global Illumination computation, one of the most important aspect of generating realistic pictures.

2.1 Basic radiometric quantities

We start by giving a couple of definitions of basic radiometric quantities, explained in detail in the book of McCluney [McC94]. They will help explain the principles of rendering, particularly the light transport equation, detailed in the next section.

Radiant energy Q is the quantity of energy propagating onto, from, or through a surface of given area in a given period of time. We consider the particle nature of light, in which radiant energy can be expressed in terms of the number of photons that are propagating onto, from, or through the surface of given area in a given period of time. Radiant energy is measured in Joules (J).

Radiant flux (or power) Φ is the rate of flow of radiant energy per unit time, i.e., the quantity of energy transferring through a surface or region of space per unit time. It is expressed in Watts (W), or Joules per second ($J.s^{-1}$), and is defined as follows:

$$\Phi = \frac{dQ}{dt} \tag{2.1}$$

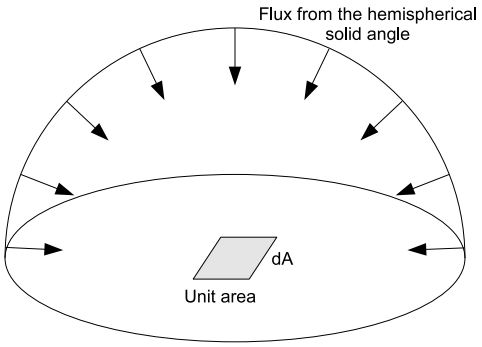


Figure 2.1: Irradiance is the radiant flux per unit area incident at a point of surface coming from a hemispherical solid angle.

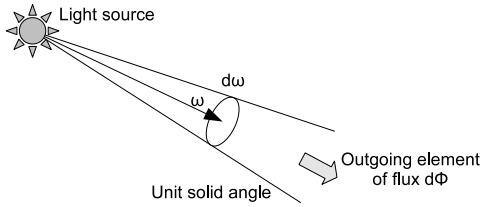


Figure 2.2: Intensity is the radiant flux per unit solid angle $d\omega$ emerging from a point.

Irradiance E is the radiant flux per unit area dA of a given surface, which is incident on or passing through a point in the given surface (Figure 2.1). All directions in the hemisphere around the surface point have to be included. It is expressed in Watts per square meter ($W.m^{-2}$) and is defined as follows:

$$E = \frac{d\Phi}{dA} \tag{2.2}$$

Radiant exitance is the same notion as irradiance, except that the energy leaves the surface rather than being incident to it. It is often written B.

Radiant intensity I is the radiant flux per unit solid angle $d\omega$, which is incident on, emerging from, or passing through a point in space in a given direction (Figure 2.2). It is expressed in Watts per steradian ($W.sr^{-1}$) and is defined as follows:

$$I = \frac{d\Phi}{d\omega} \tag{2.3}$$

Radiance L is the radiant flux per unit solid angle and per projected unit area, which is incident on, emerging from, or passing through a given point of a surface in a given direction (Figure 2.3). It is expressed in Watts per square meter per steradian ($W.m^{-2}.sr^{-1}$) and is defined as follows:

$$L = \frac{d^2\Phi}{d\omega dA_p} = \frac{d^2\Phi}{d\omega \cos\theta dA} \tag{2.4}$$

where $dA_p = \cos\theta dA$ is the projected area, area of the surface orthogonal to the solid angle

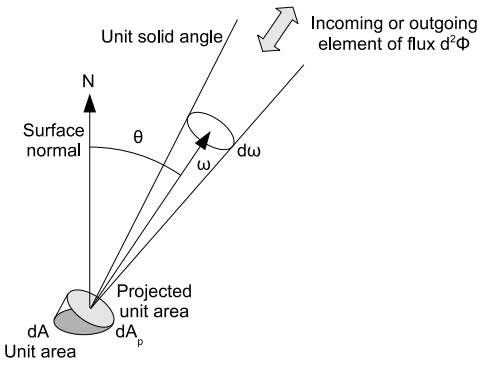


Figure 2.3: Radiance L is the radiant flux per unit solid angle $d\omega$ and per projected unit area dA_p, which is incident on, emerging from, or passing through a point of a surface in direction ω.

direction ω and projected from the original element of area dA. The angle between the surface normal and the solid angle direction is called θ.

Radiance is the main radiometric quantity that is used to describe light distribution in an environment. The other basic radiometric quantities can be defined in terms of radiance, which is integrated over a solid angle or an area.

Given incident radiance $L_i(P, \omega_i)$ from direction ω_i at a point P on a surface of normal N, we can compute the irradiance $E(P)$ at the point P by summing the contribution of radiance from every direction in the hemisphere $\Omega_+(N)$ oriented by N:

$$E(P) = \int_{\Omega_+(N)} L_i(P, \omega_i) \cos\theta_i \, d\omega_i \tag{2.5}$$

The cosine of the incidence angle, $\cos\theta_i$, is often defined as the dot product $(\omega_i \cdot N)$.

Spherical coordinates are a practical means to describe the direction vectors $\omega_i \in \Omega_+(N)$ using two angles, the azimuthal angle $\phi_i \in [0, 2\pi)$ and the elevation angle $\theta_i \in [0, \pi/2]$, which is the angle between ω_i and N. The unit solid angle is expressed as $d\omega_i = \sin\theta_i \, d\theta_i d\phi_i$. The irradiance becomes:

$$E(P) = \int_0^{2\pi} \int_0^{\pi/2} L_i(P, \phi_i, \theta_i) \cos\theta_i \sin\theta_i \, d\theta_i d\phi_i \tag{2.6}$$

If $L_i(P, \phi_i, \theta_i)$ is constant for every direction of the hemisphere and equal to $L_i(P)$, the irradiance becomes simple:

$$E(P) = \pi L_i(P) \tag{2.7}$$

Given outgoing radiance $L_o(P, \omega_o)$ in direction ω_o from a small surface around point P of normal N emitting light, we can compute the intensity $I(P_l, \omega_o)$ at the center point P_l by summing the contribution of radiance for each point of the emitting surface:

$$I(P_l, \omega_o) = \int_{A_l} L_o(P, \omega_o) \cos\theta_o \, dA \tag{2.8}$$

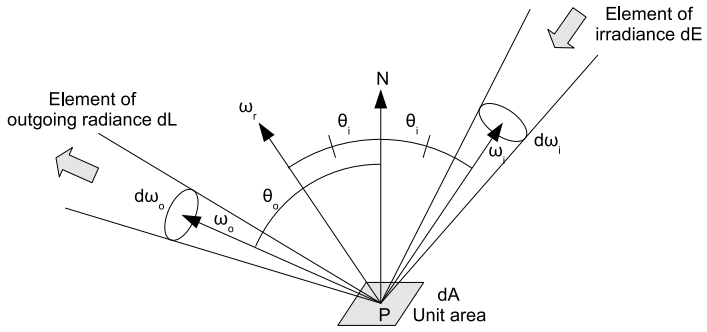


Figure 2.4: Vector and angle definitions for BRDF models. Incident light from direction ω_i is reflected on point P of surface dA to the direction ω_o. N is the normal to the surface and ω_r the ideal reflection direction.

However, intensity is normally applied to points. A_l is actually considered very small compared to the distance of the observer from the light source.

2.2 Material properties: BRDFs

Each surface in an environment has its own material properties affecting incident light scattering and reflection. In this section, we present the notion of a Bidirectional Reflectance Distribution Function BRDF as well as some BRDF models used for different types of materials.

Bidirectional Reflectance Distribution Functions (BRDFs) $f_r(P, \omega_o, \omega_i)$ give a description of reflected light distribution at a point P of a surface given an incident light flux direction ω_i and a reflection direction ω_o (Figure 2.4). These 4D functions depend on the local microgeometry. A surface material defined by a BRDF does not vary spatially, the BRDF parameters are the same everywhere on the surface. They are defined as the ratio of outgoing radiance over the incoming irradiance:

$$f_r(P, \omega_o, \omega_i) = \frac{dL_o(P, \omega_o)}{dE(P, \omega_i)} \tag{2.9}$$

BRDFs have two important properties: reciprocity and energy conservation. The reciprocity rule (Helmholtz reciprocity) gives:

$$f_r(P, \omega_o, \omega_i) = f_r(P, \omega_i, \omega_o) \tag{2.10}$$

for all pairs of directions ω_o and ω_i in $\Omega_+(N)$. The energy conservation rule requires that the total energy of reflected light is less than or equal to the energy of incident light. For every direction ω_o in $\Omega_+(N)$:

$$\int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos\theta_i \, d\omega_i \leq 1 \tag{2.11}$$

Reflectance (also known as albedo) is defined as follows:

$$\rho(\Omega_i, \Omega_o) = \frac{\int_{\Omega_o} \int_{\Omega_i} f_r(\omega_o, \omega_i) \cos\theta_o \cos\theta_i \, d\omega_i d\omega_o}{\int_{\Omega_i} \cos\theta_i \, d\omega_i} \tag{2.12}$$

where Ω_i and Ω_o can be a single direction, a solid angle or the whole hemisphere.

If Ω_i is the whole hemisphere and Ω_o a single direction, hemispherical-directional reflectance is obtained and gives the total reflection in direction ω_o due to constant illumination. It is a 2D function defined as follows:

$$\rho_{hd}(\omega_o) = \frac{1}{\pi} \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos\theta_i \, d\omega_i \tag{2.13}$$

If Ω_i and Ω_o are the whole hemisphere, the hemispherical-hemispherical reflectance is obtained and is a single constant value that represents the fraction of incident light reflected by a surface when the incident light is constant from every direction. This value is defined as:

$$\rho_{hh} = \frac{1}{\pi} \int_{\Omega_+(N)} \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos\theta_o \cos\theta_i \, d\omega_o d\omega_i \leq 1 \tag{2.14}$$

A BRDF is called isotropic when only one azimuthal angle is necessary (ϕ_i, ϕ_o, or the difference between these two angles).

Several BRDF models have been designed, used for different types of materials and for different purposes. Some of them are defined ad hoc, making them easy for a designer to configure, while others use a more physically-based approach.

Lambertian surfaces

Lambertian surfaces are perfect diffuse surfaces that scatter incident light uniformly in every direction. This model is well-adapted to matte surfaces. Is is also a very simple analytical model that simplifies many calculations related to the light transport equation. The BRDF of a Lambertian surface is defined as:

$$f_r(P, \omega_o, \omega_i) = \frac{\rho_d}{\pi} = k_d \tag{2.15}$$

where $\rho_d \leq 1$ is the diffuse albedo of the surface (hemispherical-hemispherical reflectance in this case). k_d is the diffuse BRDF and $k_d \leq 1/\pi$. The hemispherical-directional reflectance can be analytically determined:

$$\rho_{hd}(P, \omega_o) = \frac{\rho_d}{\pi} = k_d \tag{2.16}$$

Phong reflectance model

The Phong's reflectance model [Pho75] is an isotropic ad hoc model, using simple and intuitive parameters to be defined, which is adapted to plastic-like shiny materials. It is the sum of a diffuse reflection term (as for Lambertian surfaces) and a specular term. Since the Phong's

model does not respect the energy conservation rule, it has been modified [Lew94] and is then defined as follows:

$$f_r(P, \omega_o, \omega_i) = k_d + k_s \, (\omega_r \cdot \omega_o)^s \tag{2.17}$$

$$= \frac{\rho_d}{\pi} + \frac{\rho_s(s+2)}{2\pi} \, (\omega_r \cdot \omega_o)^s \tag{2.18}$$

where ρ_s is the specular albedo (with $\rho_d + \rho_s \leq 1$), k_s the specular reflectance coefficient, s is the shininess giving an information about the size of the specular highlights (the higher, the smaller are the highlights), and ω_r is the ideal reflection vector defined as follows:

$$\omega_r = 2 \, (\omega_i \cdot N) \, N - \omega_i \tag{2.19}$$

2.3 Light transport equation

The light transport equation (or rendering equation) describes the equilibrium distribution of radiance in an environment. It gives the amount of reflected (or outgoing) radiance $L_o(P, \omega_o)$ from a unit surface dA centered at point P along reflection direction ω_o given the irradiance dE from the environment at this point. It is obtained by integrating the product of the BRDF $f_r(\omega_o, \omega_i)$ with the differential irradiance dE:

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \, dE(P, \omega_i) \tag{2.20}$$

$$= L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) L_i(P, \omega_i) \cos\theta_i \, d\omega_i \tag{2.21}$$

where $L_e(P, \omega_o)$ is the radiance due to self-emission, $L_i(P, \omega_i)$ is the incident radiance and $f_r(P, \omega_o, \omega_i)$ is the BRDF.

2.3.1 Light sources

Light sources are surfaces that emit light by themselves, they are only defined by their outgoing radiance. The outgoing radiance $L_o(P_l, \omega_o)$ of a point P_l of a light source is then defined as:

$$L_o(P_l, \omega_o) = L_e(P_l, \omega_o) \tag{2.22}$$

If there is no participating media causing scattering along the ray $P_l P$, the radiance is constant along this ray. Therefore, we can modify the expression of the irradiance (Equation 2.5) and the expression of the light transport equation (Equation 2.21) using the following relation:

$$L_i(P, \omega_i) = L_o(P_l, -\omega_i) \tag{2.23}$$

Area light sources

An area light source is a surface that emits light. To account for its contribution in the irradiance and light transport equations, it is easier to integrate over the light source area rather than the subtended solid angle. In the previous section, irradiance and light transport equations

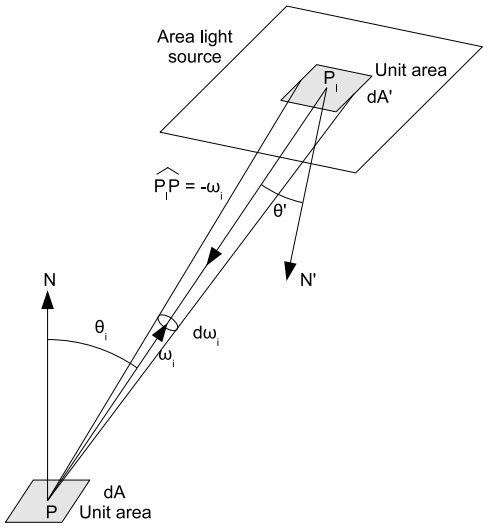


Figure 2.5: Vector and angle definitions for an area light source. Incident light from direction ω_i in unit solid angle $d\omega_i$ is emitted from the unit area dA' at distance $\|PP_l\|$ from the surface centered in P. N' is the normal to the area light surface.

are defined using the differential solid angle $d\omega_i$. We can express this solid angle using the unit area dA' using the following expression:

$$d\omega_i = \frac{\cos\theta' \, dA'}{\|PP_l\|^2} \tag{2.24}$$

where dA' is a unit area of the light source, θ' is the angle between the light source surface normal N' and the vector from the light source to the surface being lit, $\widehat{P_lP} = -\omega_i$ (Figure 2.5).

The irradiance at point P due to this light source becomes

$$E(P) = \int_{A'} L_i(P, \omega_i) \cos\theta_i \frac{\cos\theta' \, dA'}{\|PP_l\|^2} = \int_{A'} L_o(P_l, -\omega_i) \cos\theta_i \frac{\cos\theta' \, dA'}{\|PP_l\|^2} \tag{2.25}$$

and the light transport equation becomes

$$\begin{aligned} L_o(P, \omega_o) &= L_e(P, \omega_o) + \int_{A'} f_r(P, \omega_o, \omega_i) L_i(P, \omega_i) \cos\theta_i \frac{\cos\theta' \, dA'}{\|PP_l\|^2} \\ &= L_e(P, \omega_o) + \int_{A'} f_r(P, \omega_o, \omega_i) L_o(P_l, -\omega_i) \cos\theta_i \frac{\cos\theta' \, dA'}{\|PP_l\|^2} \end{aligned} \tag{2.26}$$

with A' the area of the light source surface.

Omnidirectional point light sources

Point light sources are light emitters with a zero area. They are a practical way to approximate very small light sources since they greatly simplify the light transport equation evaluation.

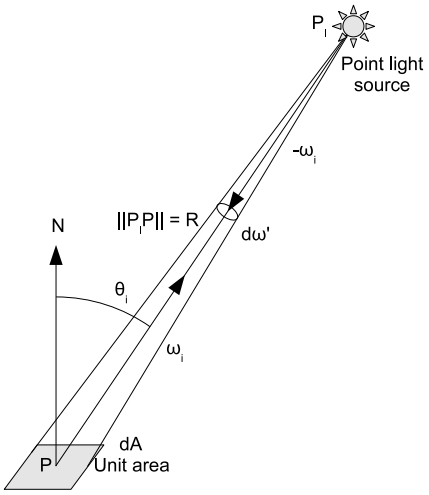


Figure 2.6: Vector and angle definitions for a point light source. The flux received by dA due to the point light source at P_l, in direction ω_i, at distance R, is the flux emitted by the light source in the unit solid angle $d\omega'$.

Omnidirectional point light sources emit a constant flux in every direction. A unit area dA centered in P receives the flux emitted by the light source in the unit solid angle $d\omega'$ starting from the point light source position P_l and subtended by dA. Since the radiant intensity I represents the flux emitted by the light source per unit solid angle, the irradiance of the unit area dA due to this light source is

$$E(P) = \frac{I \cos\theta_i}{R^2} \tag{2.27}$$

Since flux is received by dA in only one direction, $\omega_i = \widehat{PP_l}$, the light transport equation (Equation 2.20) becomes

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i)\delta(\omega_i - \widehat{PP_l})\, dE(P, \omega_i) \tag{2.28}$$

where $\delta(x)$ is the dirac operator defined as follows:

$$\delta(x) = 0 \qquad \text{if} \quad x \neq 0 \tag{2.29}$$

$$\int_D \delta(c - x) f(x)\, dx = f(c) \qquad \text{if} \quad c \in D \tag{2.30}$$

The light transport equation then becomes simple to evaluate:

$$L_o(P, \omega_o) = L_e(P, \omega_o) + f_r(P, \omega_o, \widehat{PP_l}) \frac{I \cos\theta_i}{R^2} \tag{2.31}$$

Directional light sources

A directional light source can be represented as a collimated beam. In this case, light reaches surfaces in parallel beams. Irradiance E can be expressed for any virtual surface inside the beam. If the normal N of a surface makes an angle θ_l with the beam direction ω_l, the irradiance $E(P)$ at any point P of this surface is $E(P) = E_l \cos\theta_l$ with E_l the irradiance associated with the light beam. In this case, the light transport equation becomes

$$L_o(P, \omega_o) = L_e(P, \omega_o) + f_r(P, \omega_o, \omega_l) E_l \cos\theta_l \tag{2.32}$$

2.3.2 Visibility function and ambient occlusion

In previous sections, the light transport equation was evaluated considering no obstacle between light sources and the receiver surface. However, in normal scenes obstacles are always present, reducing the contribution of light sources for a set of incidence directions, hence creating shadows. We rewrite the light transport equation to take this shadowing into account:

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) V(P, \omega_i) L_i(P, \omega_i) \cos\theta_i \, d\omega_i \tag{2.33}$$

where $V(P, \omega_i)$ is the visibility function, equal to 1 when a light source is visible and equal to 0 when an obstacle prevents light from reaching the surface unit area.

The integral of the light transport equation is usually expensive to compute. Computing the visibility is usually the bottleneck of the rendering algorithms.

2.4 Global illumination

The previous sections considered surface lighting only due to light sources with or without occlusion. Such computations are called *direct lighting*. However, surfaces in any scene are illuminated not only by light sources, but also by light bouncing on neighbor surfaces. The contribution of neighbor surfaces is important for realism in rendered scenes, but is very computationally expensive: if N surface unit areas are present in a scene (N is large), $N(N-1)$ interactions between surfaces have to be considered for a single bounce of light.

As mentioned earlier, in the absence of any participating medium in the scene, radiance along a ray is constant. This means that the outgoing radiance at a point of a surface in a given direction contributes to the irradiance of a point of another surface. For a given surface of the scene, all other surfaces are then considered as light sources. The outgoing radiance from these surfaces is actually due to the incoming radiance from the light sources and the other surfaces. This shows that the relation is recursive.

The light transport equation (Equation 2.21) can be written a different way:

$$L = L_e + \mathcal{T}.L \tag{2.34}$$

where L is the equilibrium light in the scene, L_e is the light due to emission of the light sources and $\mathcal{T}$ is the light transport operator. Solving Equation 2.34 for L gives:

$$L = L_e(1 - \mathcal{T})^{-1} \tag{2.35}$$

Expansion of this equation to a Neuman series gives:

$$L = L_e + \mathcal{T}.L_e + \mathcal{T}^2.L_e + \mathcal{T}^3.L_e + \dots \tag{2.36}$$

$\mathcal{T}.L_e = L_{direct}$ represents the first bounce of light after leaving the light sources. The previous equation can be rewritten in term of direct lighting:

$$L = L_e + L_{direct} + (\mathcal{T}.L_{direct} + \mathcal{T}^2.L_{direct} + \dots) \tag{2.37}$$

The content of the parenthesis represents the second and higher order bounces of light, it is considered as $L_{indirect}$. This part is usually the most expensive term to evaluate. In some scenes, the evaluation of the first bounce of indirect light is enough ($\mathcal{T}.L_{direct}$), the next bounces only affecting the result slightly. However, this does not hold for scenes with complex lighting interactions and for physically based rendering. Simulating the dispersion of the daylight into a building may require to compute a solution taking into account multiple light bounces. Most part of the real-time and/or interactive global illumination techniques only account for the first bounce.

2.5 Light paths notation

The different types of light paths are usually characterized using the regular expression notation proposed by Paul Heckbert [Hec90]. The eye is denoted by E, the light source by L, a diffuse surface by D, a specular one by S and '*' means "zero or more" whereas '+' means "once or more". For example, LD*E describes a path where light has been emitted by a light source, then has bounced over several diffuse surfaces before reaching the eye. A LD*SE path describes light bouncing on several diffuse surfaces then on a mirror before reaching the eye. Most part of the time, the various types of light paths are computed using several different methods, like photon mapping for the caustics (LS+DE), and irradiance cache for the diffuse inter-reflections (LD*E).

Chapter 3

Related Work

In this chapter, we recall the fundamentals of global illumination, and detail significant previous contributions. As our work strongly relies on photon mapping and irradiance caching, we detail both these methods. We also present Monte Carlo integration and the associated variance reduction techniques. Finally, we present an overview of our contributions compared to previous approaches.

3.1 Global illumination

As introduced in the previous chapter, Global illumination accounts for the multiple interactions between light and matter within a scene. The book by Pharr and Humphreys [PH04] provides both theoretical and practical details on rendering and global illumination. In [Gla94], Glassner details the physics and algorithms needed for accurate, physically-based rendering. Advanced topics on global illumination can be found in [DBB02].

As explained in the previous section, the aim of global illumination is the resolution of the rendering equation [Kaj86], also known as the light transport equation (Section 2.3). This integral equation expresses the outgoing radiance L_o at a given point x as a function of the self outgoing radiance L_e and the radiance L_i incoming from locations visible from x:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_o, \omega_i) L_i(x, \omega_i) \cos\theta_i \mathrm{d}\omega_i \tag{3.1}$$

where f_r is the Bidirectional Reflectance Distribution Function (BRDF), representing how the incoming lighting is reflected by the surface a point x. Unfortunately, this equation cannot be solved analytically in the general case. Therefore, many numerical methods have been proposed. In the next section we present the most significant approaches.

3.2 Lighting simulation techniques

3.2.1 Radiosity

The radiosity method was originally introduced by Goral *et al.*[GTGB84]. This approach aims at computing diffuse interreflections by using a finite elements method: each surface of

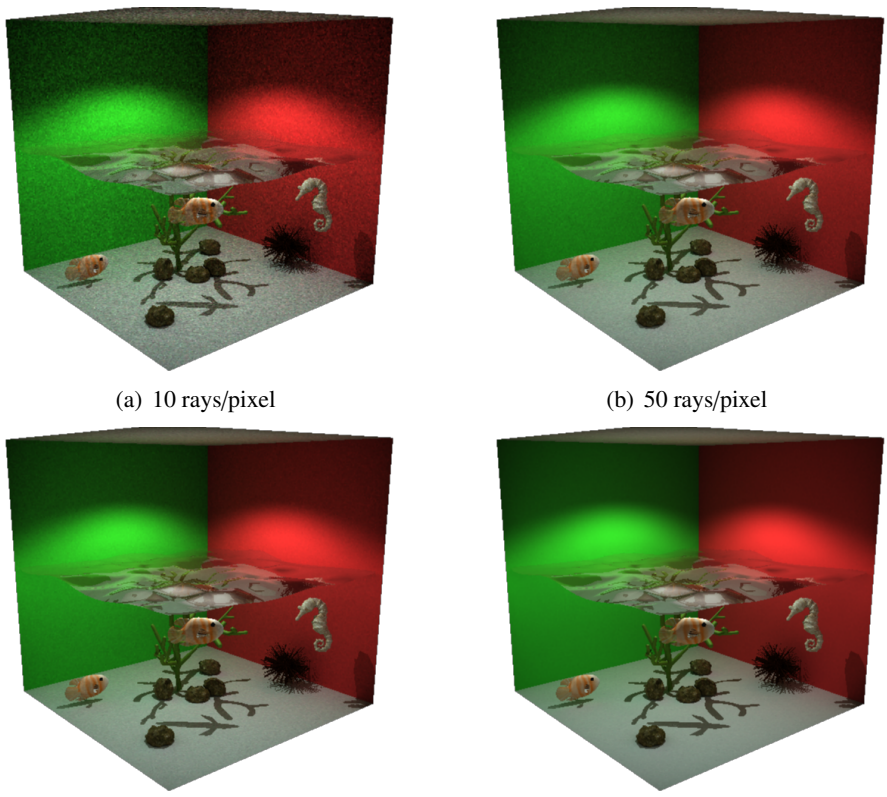
the considered scene (including light sources) is subdivided into patches, for which the lighting is assumed to be constant. The radiosity method consists in calculating the light transfer between patches. Even though this method provides high quality results, it requires finely subdivided surfaces, hence high computation time. Moreover, the use of finite elements can lead to meshing artifacts. Many methods have been proposed for adaptive subdivision, yielding finely tessellated surfaces only in the regions containing discontinuities (discontinuity meshing) thus reducing the rendering time at the cost of high memory consumption. However, high quality results can only be obtained through the use of costly final gathering [SSS01, SSS02]. Once the global illumination solution is computed, the result can be stored in light maps and displayed in real-time using graphics hardware. Many computer games rely on this method to render globally illuminated scenes at the cost of simple texturing.

3.2.2 Monte Carlo path tracing

Monte Carlo integration [PTVF88] is one of the most commonly used method for solving the rendering equation. This method can simulate various global illumination effects, such as diffuse and glossy interreflections. Monte Carlo path tracing follows the reverse path of light: rays are traced from the observer to the scene elements. For each visible point, the rendering equation is solved numerically by Monte Carlo integration: the lighting at a given point is obtained by tracing random rays from this point towards the scene. Note that for two distinct visible points the computation of the lighting is completely independent, making this method easily parallelizable and inexpensive in terms of memory. However, the incoming radiances of adjacent pixels of the final image may be uncorrelated, yielding noise artifacts (Figure 3.1). Due to the slow convergence of Monte Carlo methods, high quality rendering requires to trace many rays, yielding a significant computational cost while only reducing the noise [VG95]. Several methods have been proposed to reduce the variance (see Section 3.3) inherent to Monte Carlo estimators. Even though these methods generally reduce the number of rays to be cast and/or improve the rendering quality, the cost of Monte Carlo path tracing remains high. Furthermore, this method cannot account for caustics, which require tracing rays from the light sources to the objects. A variant of the algorithm, Bidirectional Path Tracing [LW93], also cast rays from the light sources to the scenes. These rays are connected to the rays cast from the observer to form light paths, allowing to simulate caustics and other effects. However, photon mapping is usually the method of choice when it comes to simulate complex lighting effects.

3.2.3 Photon mapping

Photon mapping [Jen96, Jen01] is based on the computation of the real path of the light, that is from light sources to the objects of the scene. The flux of each light source is split in a number of outgoing rays, carrying *photons*. When a ray intersects an object, the energy and incoming direction of the photon are stored in a *k*d-tree [Ben75], called *photon map*. Then, the photon is reflected in a random direction depending on the surface BRDF. Once the photons are stored in the photon map, the indirect illumination at a given point x can be computed by gathering the photon hits in a close neighborhood around x. Since the computed light path



(a) 10 rays/pixel (b) 50 rays/pixel

(c) 100 rays/pixel (d) 500 rays/pixel

Figure 3.1: Monte Carlo path tracing: the accuracy of the method comes at the expense of computational cost. Images: Pascal Gautron.

corresponds to the actual light path, this method accounts for complex lighting effects such as caustics and participating media. Many approaches attempt to optimize the construction and rendering of the photon map, such as [SW00, Chr99, FMH05, HP02, KW00, Sch03, WGS04].

However, photon mapping is a stochastic process which can introduce bias and variance in the reconstructed indirect illumination: a rendered image exhibits either noise or blur. Using a direct visualization of the photon map, the reconstructed indirect illumination is generally very noisy due to the limited number of photons available around the visible points (Figure 3.5). An additional computation step called *final gathering* is often used for high-quality. Using several hundreds of additional rays for each visible point, depending on the resolution of the image and the number of cast rays, final gathering can take up to several hours for a single image.

As our proposed Photon Driven Irradiance Cache method strongly relies on photon mapping, we detail the method in the following sections.

3.2.3.1 First pass: photon tracing

During the first pass, *photons* are emitted from the light sources of the scene. These light sources can be typical computer graphics light sources such as point lights, directional lights and area light sources, or they can be physically based sources with arbitrary geometry and distributions. Any type of light source can be used.

Photon mapping typically requires a large amount of photons, just as in nature. The power of the light sources is divided among all the emitted photons: each photon p_i carry a fraction of the light source power, a *flux* value Φ_i. Photon tracing works exactly the same way as ray tracing, except that photons propagate flux whereas rays gather radiance. When a photon hit an object, it can be reflected, transmitted or absorbed, depending on the material parameters of the surface of the object. For each hit on a diffuse surface, the photon is stored in a global data structure called *photon map* (Figure 3.2). Note that each emitted photon can be stored several times along its path. The photons carry the following information:

- x, the position of the hit,
- $\vec{n}$, the normal of the surface at x,
- $\vec{\omega_i}$, the incident direction of the photon,
- Φ, the flux carried by the photon.

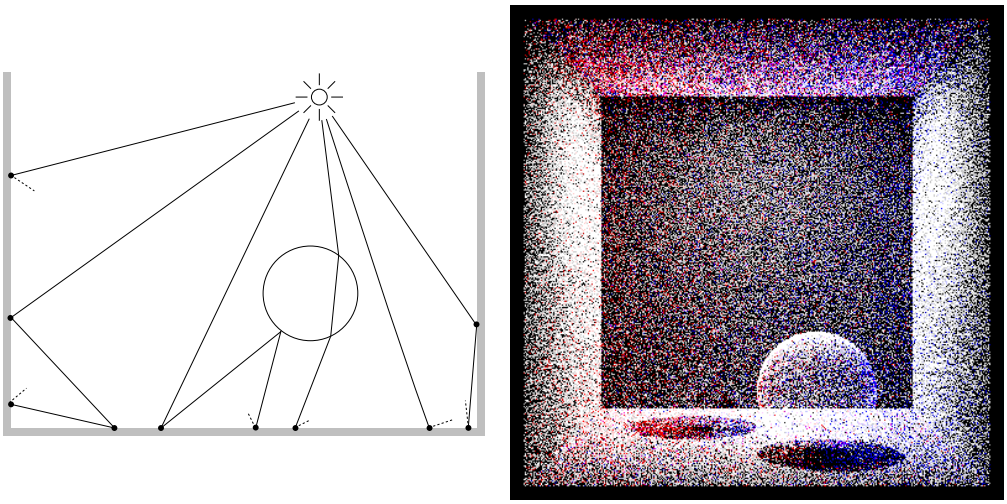
The photon map is a representation of all the stored photons in the model and is decoupled from the geometry of the scene. Note that for a static scene, the computed photon map is view-independent and the same photon map can be used to render different viewpoints of the scene.

Rendering with a photon map relies on estimating the photon density at several points in space. Hence, the global data structure used for the photon map needs to allow efficient nearest neighbors search in a three-dimensional point set. This is why the *k*d-tree data structure is usually used. As the nearest neighbors search is more efficient when the *k*d-tree is balanced, the construction of the *k*d-tree is usually performed after all the photons have been emitted, at the end of the photon tracing pass. The resulting *k*d-tree is balanced, and yields fast nearest neighbors search, associated with a compact representation.

3.2.3.2 Second pass: rendering from the photon map

During the second pass, the photon map is used to estimate the reflected radiance incoming from visible points on diffuse surfaces. The density of the photons at a point indicates how much light the region receives. From the photon density, one can deduce the incoming irradiance at a point, then the radiance reflected by a diffuse surface. A standard recursive ray tracing procedure is used for computing the radiance reflected from specular surfaces.

Density estimation The photon map represents incoming flux on the surfaces of the scene. Each photon transports a fraction of the light source power, and a photon hit in a region indicates that this region is receiving some illumination from the light source, either directly or indirectly. However, based on a single photon we cannot say how much light the region receives. This is given by the *photon density*, $\frac{d\Phi}{dA}$. Therefore, to estimate the irradiance for a given region we need to compute the density of the neighbor photons. Density estimation is



(a) photon tracing (b) photon visualization

Figure 3.2: The photon map is built using photon tracing in which photons are emitted from the light sources and stored as they interact with the diffuse surfaces in the model.

still a research area in statistics, and several books have been written about density estimation problems ([Sil86, Sim98]). It is well known that the histogram-based density estimation approach is inferior to another class of density estimations approaches: the *kernel density estimation* techniques. Kernel density estimation techniques operate directly on the individual elements, in our case the photon hits, and use a local kernel operator to smooth the estimate.

The radiance estimate We want to compute the reflected radiance L_r at a visible point x in direction $\vec{\omega}$, given by:

$$L_r(x,\vec{\omega}) = \int_\Omega f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}')(\vec{n_x}\cdot\vec{\omega}')d\vec{\omega}' \tag{3.2}$$

where Ω is the hemisphere above x, f_r is the BRDF at x, L_i the incoming radiance in every direction and n_x the normal at x. To solve this integral we need information about L_i, but the photon map provides only information about the incoming flux Φ_i. Using the relationship between radiance and flux:

$$L_i(x,\vec{\omega}) = \frac{d^2\Phi_i(x,\vec{\omega}')}{(\vec{n_x}\cdot\vec{\omega}')d\vec{\omega}'dA_i}\,, \tag{3.3}$$

where dA_i is the differential unit area, the integral can be rewrote as:

$$L_r(x,\vec{\omega}) = \int_\Omega f_r(x,\vec{\omega}',\vec{\omega})\frac{d^2\Phi_i(x,\vec{\omega}')}{dA_i}\,. \tag{3.4}$$

The incoming flux Φ_i is approximated using the photon map by locating the n photons that have the shortest distance to x. Each photon p has the power $\Delta\Phi_p(\vec{w_p})$ and, by assuming that

the photon intersects the surface at x, we obtain:

$$L_r(x, \vec{\omega}) \approx \sum_{p=1}^{n} f_r(x, \vec{\omega_p}, \vec{\omega}) \frac{\Delta\Phi_p(x, \vec{w_p})}{\Delta A} \tag{3.5}$$

This is computed by expanding a sphere around x until it contains n photons then using these n photons to estimate the radiance (Figure 3.3). The value ΔA is computed by assuming that the surface is locally flat around x, then projecting the sphere onto the surface and using the area of the resulting circle:

$$\Delta A = \pi r^2 \,, \tag{3.6}$$

where r is the radius of the search sphere, i.e. the largest distance between x and each of the photons.

When x is a visible point on a diffuse surface, the radiance estimate is:

$$L_r(x, \vec{\omega}) \approx \frac{1}{\pi r^2} \frac{\rho_d}{\pi} \sum_{p=1}^{n} \Delta\Phi_p(x, \vec{w_p}) \,, \tag{3.7}$$

where ρ_d is the diffuse reflectivity of the surface at x and r is the distance to the n-nearest neighbor photon. This process is equivalent of using a *box kernel* density estimation technique.

Bias and variance However, the radiance estimate (Equation 3.7) exhibits strong variance: the direct visualization of the photon map yields very noisy pictures (Figure 3.5). The variance of the estimator can be reduced by storing more photons in the photon map. One can also use a greater number n of photons during the radiance estimate, but in this case variance is traded for bias in the estimate. Finally, the contribution of each photon to the radiance estimate can also be filtered: the further the photon is from the point x, the less it contributes to the radiance estimate. This is equivalent to performing density estimation with a different kernel (cone kernel, Gaussian kernel, etc. . .).

Another problem that may arise is called *boundary bias*. When estimating irradiance close to a wall or close to the boundary of a surface, the projected area ΔA of the search sphere may not correctly represent the true area covered by the photons (Figure 3.4). The value of ΔA is overestimated and the radiance estimate is darker than it should be. A similar problem arises on curved surfaces, because of the assumption that the surface around x is planar: in this case, ΔA is underestimated. A more accurate estimation of ΔA could be computed, by using the convex hull of the nearest photons for example, but doing it for every radiance estimate is too costly.

3.2.3.3 Final gathering

To produce high-quality pictures, a costly additional pass called *final gathering* is performed. The final gathering step consists in computing the outgoing radiance of each visible point x by solving the equation:

$$L_o(x, \omega_o) = \int_{\Omega} f_r(x, \omega_o, \omega_i) L_i(x, \omega_i) \cos\theta_i \mathrm{d}\omega_i \tag{3.8}$$

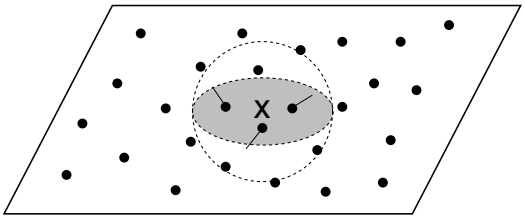
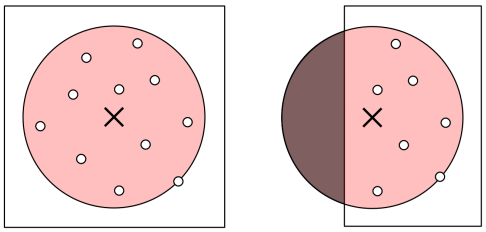
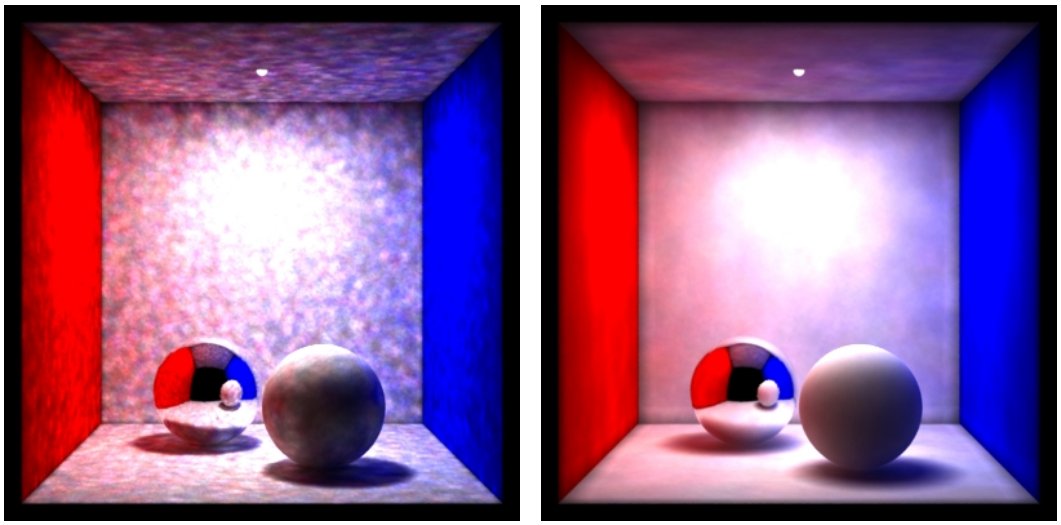


Figure 3.3: Reflected radiance is evaluated by locating the nearest photons in the photon map in order to estimate the local photon density. This approach can be seen as expanding a sphere around the intersection point until it contains enough photons.



(a) Area estimation (b) Boundary bias

Figure 3.4: *(a)* The photon density is estimated based on the surface area covered by the sphere. *(b)* This can lead to bias in estimate, close to the geometry boundaries.



(a) 50 nearest photons (b) 500 nearest photons

Figure 3.5: Indirect lighting only. Direct visualization of the photon map yields noisy pictures. *(a)* 50 photons are used to estimate radiance. *(b)* 500 photons are used to estimate radiance. Noise is reduced, at the expanse of a larger bias near boundaries.

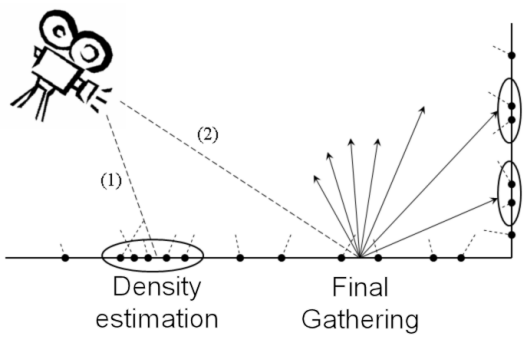


Figure 3.6: During the final gathering pass, secondary rays are cast from each visible point. Contribution along each ray is computed by performing density estimation at the intersection point.

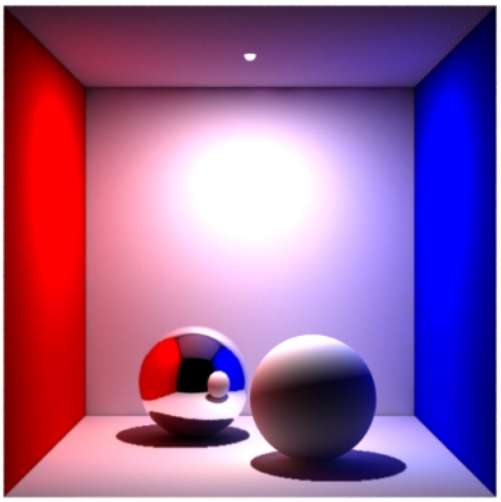


Figure 3.7: Image computed with final gathering. The rendering cost is several hundred times much higher than direct visualization of the photon map.

This equation represents the outgoing radiance at point x and direction ω_o in terms of the radiance incoming from all the directions of the hemisphere above x. Those incoming radiances can be provided by a rough estimate of the global illumination solution at points visible from point x. Namely, the final gathering gathers the incoming radiances to reconstruct the indirect lighting at the points visible from the camera. The outgoing radiance (Equation 3.8) is computed by Monte Carlo integration for each pixel of the final image. The incoming radiance along each ray is computed by performing a density estimation at the corresponding intersection point (Figure 3.6). Moreover, high quality results require tracing many rays for each visible point, yielding a prohibitive computational cost. Nevertheless, this costly additional step provides high quality images even with a very coarse global illumination solution (Figure 3.7). Note that final gathering is inherently view-dependent and need to be performed for each viewpoint.

Several methods have been proposed to speed-up the final gathering step. In [Chr99], during a preprocessing step, an approximate value of irradiance at each photon location is computed using density estimation and stored in its associated data structure. When performing final gathering, the contribution of each cast ray is computed using the irradiance value of the nearest photon to the intersection point, rather than using density estimation. This method considerably reduces the number of density estimations that are performed, by taking advantage that many final gathering rays will intersect the same neighborhood of a photon. In [HHS05], Havran et al. explain how to reverse the final gathering pass to exploit logarithmic behavior of *k*d-trees. In classical photon mapping, the number of final gathering rays can grow up to

several hundred of millions, whereas the typical number of stored photon is about few millions. By reversing the final gathering pass, a *k*d-tree is built over all the final gathering rays hit, and density estimation is performed for each stored photon. However, accurately performing the final gathering for each pixel is still computationally expensive. The irradiance caching algorithm described hereafter performs the final gathering step only for a sparse set of visible points in the scene, hence reducing the rendering cost. Although final gathering remains very costly, practically it is almost always used as it provides artifact-free images.

3.2.4 Irradiance caching

This method is the core of the Radiance imaging system [War94]. The principle underlying irradiance caching is the fact that "indirect illuminance changes slowly over a surface" [WRC88]. The irradiance caching algorithm is based on sparse sampling and spatial interpolation of indirect lighting: the indirect diffuse lighting is computed accurately at a small number of visible points using classical Monte Carlo integration. These values, called *irradiance records*, are stored in the *irradiance cache*.

Indirect lighting at other visible points is then inexpensively and accurately computed by extrapolating and interpolating the values of the records. In our work, we mainly build on the irradiance caching algorithm, which is therefore precisely detailed in this section. The irradiance caching algorithm aims at computing the indirect diffuse lighting. In this case, the BRDF $f_r(x, \omega_o, \omega_i)$ can be replaced by the diffuse reflectance of the surface $\rho_d(x)$:

$$L_o(x, \omega_o) = \rho_d(x) \int_\Omega L_i(x, \omega_i) cos(\theta_i) d\omega_i \tag{3.9}$$

$$= \rho_d(x) E_i(x) \tag{3.10}$$

where $E_i(x)$ is the irradiance at point x. As shown in this equation, the radiance outgoing from point x in direction ω_o does not depend on ω_o. Therefore, the irradiance E_i is sufficient for representing indirect diffuse lighting.

As explained above, the irradiance caching algorithm is based on sparse sampling and interpolation of indirect lighting. However, even though the indirect lighting changes slowly, the change rate of the lighting at a given point is highly dependent of the surrounding geometry. Ward et al. [WRC88] propose an adaptive method for deciding whether a record contributes to the indirect lighting of a neighboring point. The authors calculate an upper bound of the change rate of incoming radiance with respect to rotation and translation. For a record k located at point x_k with normal n_k, the weight at point x with normal n is the inverse of this change rate:

$$w_k(x) = \left(\frac{\|x - x_k\|}{R_k} - \sqrt{1 - n \cdot n_k} \right)^{-1} \tag{3.11}$$

where R_k is the harmonic mean distance of objects visible from point p_k. The set of records contributing to the indirect lighting at point x is:

$$S(x) = \left\{ k : w_k(x) > \frac{1}{a} \right\} \tag{3.12}$$

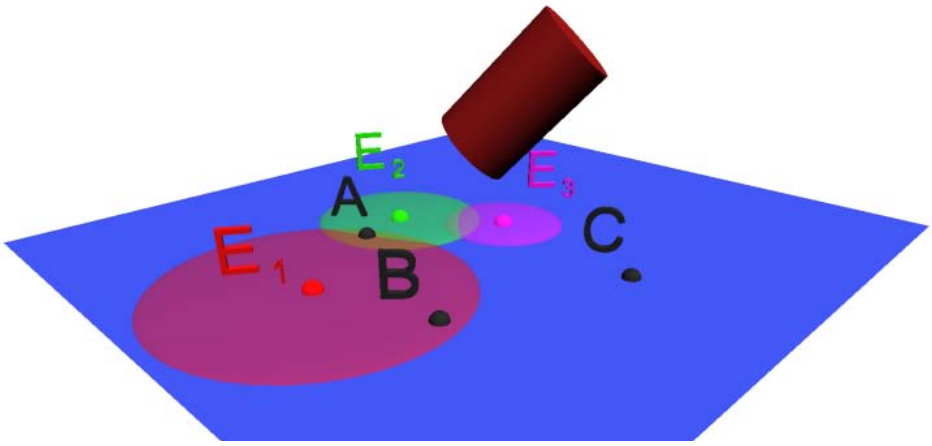


Figure 3.8: The indirect lighting of points near the irradiance records E_1, E_2, E_3 can be extrapolated and interpolated from the values computed at the location of the records. During the rendering process, three situations can happen: point A is in the zone of influence of records E_1 and E_2. In this case, the indirect lighting is interpolated from the values obtained from those records. Point B is located in the zone of influence of E_1 only, and hence its indirect lighting is extrapolated from E_1. Since no records can contribute to the indirect lighting at point C, a new record will be generated at C. Note that the size of the contribution zone of each record is inversely proportional to the distance of the surrounding cylinder.

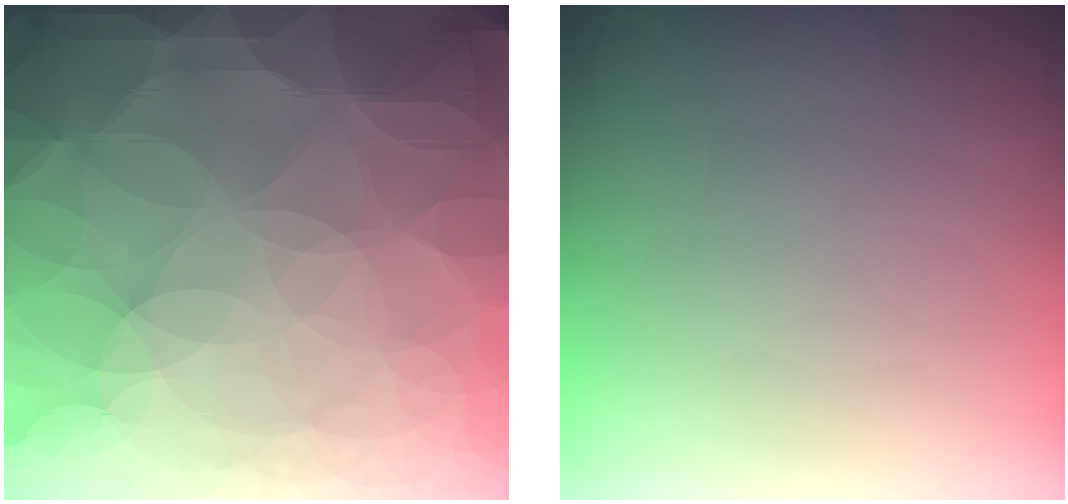
where a is a user-defined accuracy criterion. On a flat surface, each record contributes to the indirect lighting of points within a circle (Figure 3.8). The size of the influence zone of a record is constrained by R_k: the closer the surrounding objects, the smaller the zone. Even though the change of indirect lighting within the influence zone of a record is small, this change cannot be completely neglected (Figure 3.9). Consequently, Ward et al. [WH92] propose irradiance gradients, which represent how the indirect lighting changes around the location of the records. The indirect lighting at a given point x within the contribution zone of record k is extrapolated as follows:

$$E_k(x) = E_k(1 + (n_k \times n)\nabla_r + (x - x_k)\nabla_t) \tag{3.13}$$

where ∇_r and ∇_t are respectively the rotational and translational gradients, accounting for the change of lighting after rotation and translation. Krivanek *et al.* [KGBP05] proposed a more accurate method to compute theses gradients. Since several records may contribute to the indirect lighting of a single point x, Ward *et al.* [WRC88] proposes the following interpolation formula:

$$E(x) = \frac{\sum_{k \in S(x)} E_k(x) w_k(x)}{\sum_{k \in S(x)} w_k(x)} \tag{3.14}$$

The overall algorithm for irradiance caching is described in Algorithm 1. For each visible point x, the irradiance cache is queried. The sum of the weights of the neighboring records is checked against the user-defined accuracy value a (Equation 3.15). If the contributions of existing records is not sufficient (or if no records are present), an irradiance record is created



(a) no gradients (b) with gradients

Figure 3.9: The change of indirect lighting within the influence zone of a record can be significant. *(a)* Irradiance is interpolated from the records' irradiance value only. *(b)* Irradiance is interpolated using irradiance value and irradiance gradients.

at the point of interest, and stored in the irradiance cache (represented by an octree). Otherwise, existing records are used to obtain an estimate of the indirect lighting by evaluating Equation 3.14.

$$\sum_{k \in S(x)} w_k(x) \begin{cases} < a & \text{New record is required} \\ \geq a & \text{Interpolation can provide a satisfying result} \end{cases} \tag{3.15}$$

Algorithm 1: Irradiance caching

```
forall the visible point x do
    Query the irradiance cache
    if enough contributing records are found then
        Estimate indirect lighting by interpolation
    else
        Compute a record at point x
        Store the record in the cache
        Use the record to estimate the indirect lighting at x
    endif
endfall
```

Krivanek *et al.* [KGPB05] extended the irradiance caching algorithm to radiance caching. Whereas irradiance caching stores only irradiance in the records, radiance caching stores a spherical harmonics representation of the incoming radiance function. This allows to interpolate outgoing radiance for moderate glossy surfaces. Gautron *et al.* [GKBP05] proposed

a method to interactively render a radiance cache using graphics hardware, called *radiance cache splatting*. Note that the irradiance cache can also be rendered interactively with the same method.

The creation of the irradiance record is driven by the current viewpoint: the algorithm computes all the records needed to interpolate irradiance at each visible point. Therefore, in a static scene, images from other viewpoints can be generated quickly by reusing the previously created records, instead of recomputing a novel global illumination from scratch. Note that several methods can be used to compute the irradiance value during the irradiance record creation. One could compute the irradiance using classical Monte Carlo, with a final gathering from a photon map or from virtual point light sources distributed in the scene. Figure 3.10 shows a rendering of the Sponza scene using irradiance caching.

3.3 Variance reduction techniques for Monte Carlo algorithms

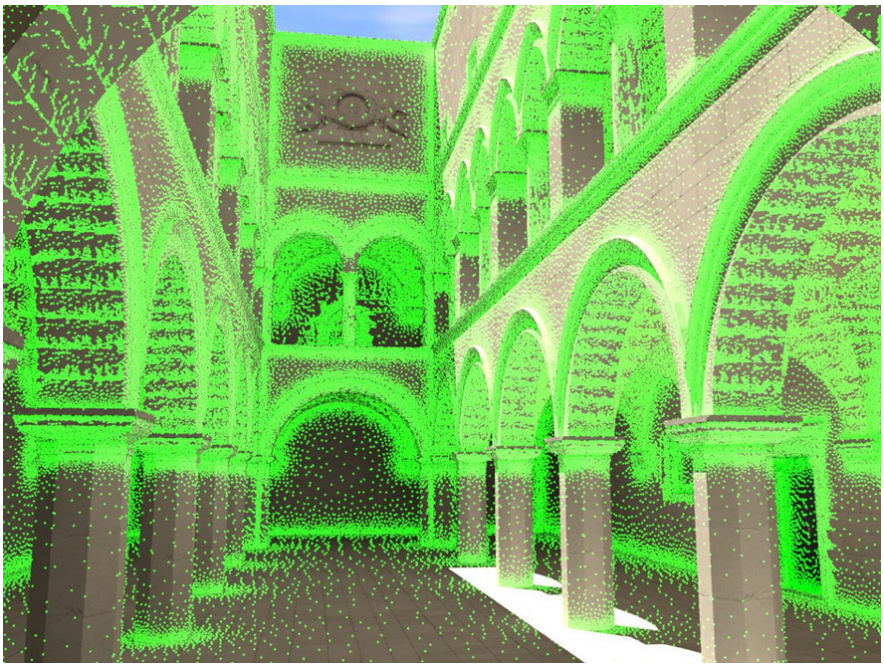
Most parts of the global illumination algorithms are based on Monte Carlo integration: path tracing, distributed raytracing, photon mapping, irradiance caching and so on. In computer graphics, Monte Carlo is generally used to numerically compute integral values. Despite being very costly, Monte Carlo based algorithms are widely used because they yield correct (i.e. unbiased) renderings. This property is essential in several domains, especially in predictive rendering and prototyping (physically based lighting simulation, architectural design, car design, etc...).

All Monte Carlo based algorithm exhibit noise at some level, and require a high computation cost to achieve high quality rendering. This is due to the fact that Monte Carlo estimators, while generally unbiased, exhibit high variance. Variance can be reduced by using more samples, which for example translates to casting more rays. In order to reduce the noise below a perceptual threshold, one has to use a huge, and often prohibitive, number of samples. Many solutions have been proposed to reduce the inherent variance of Monte Carlo methods. A good survey can be found in [Vea98].

Variance reduction can be viewed as a mean to use known information about the integration problem. If nothing is known, variance reduction cannot be achieved. At the other extreme, that is, a complete knowledge of the integrand, the variance is equal to zero and there is no need for simulation. As the second part of this thesis proposes a new approach to the problem of variance reduction for global illumination, we briefly present some of the existing methods.

We can split the variance reduction algorithm into two groups. The algorithms of the first group rely on a prior knowledge on the properties of the signal to reduce the variance (importance sampling, control variates, etc...). As for those of the second group, they modify the Monte Carlo process during the computation, taking into account the information brought by each new sample (adaptive sampling, Metropolis light transport, etc...). Our Bayesian Monte Carlo approach lies in the first group since we model the signal using a prior probabilistic model, which allows us to compute a better estimation of the integral.

All the variance reduction methods presented hereafter take advantage of a certain knowledge about the integrand. Stratified sampling and Quasi Monte Carlo methods rely on the fact that the function is well-behaved and reasonably smooth. Importance sampling requires to



(a) record positions

(b) final render

Figure 3.10: Rendering of Sponza scene using irradiance caching. Images: Arikan, O., Forsyth, D.A, O'Brien, J.F.

choose a *pdf* which has the same shape as the integrand. Control variates method exploits a crude approximation of the integrand, whose integral value is known. As explained later, our method is based on Bayesian Monte Carlo. Bayesian Monte Carlo allows us to exploit a statistical knowledge about the integrand, expressed in the form of a covariance function associated with a probabilistic model of the integrand.

3.3.1 Simple Monte Carlo

We consider the problem of computing the integral I of the product of a function $f(\mathbf{x})$ with a probability density $p(\mathbf{x})$, both functions defined on the D dimensional unit cube $[0,1)^D$:

$$I = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \qquad \text{with } \mathbf{x} \in [0,1)^D \tag{3.16}$$

Here and elsewhere, integrals without explicit range are understood to be over $[0,1)^D$. This can be assumed without loss of generality as translation to this form is always possible. Typically, $p(\mathbf{x})$ is known under its analytical form whereas $f(\mathbf{x})$ can only be evaluated numerically or through computer simulation. The basic form of Monte Carlo (MC) simulates random vectors $\mathbf{X}_1, \ldots, \mathbf{X}_n$ drawn from $p(\mathbf{x})$. Then the Monte Carlo estimate of I is:

$$\hat{I}_p = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_i)$$

It is elementary that $\hat{I}_p$ is an unbiased estimator of I since $\mathrm{E}(\hat{I}_p) = I$, and its mean square error is:

$$\mathrm{E}\left[(\hat{I}_p - I)^2\right] = \mathrm{Var}(\hat{I}_p) = \frac{\mathrm{Var}\left[f(\mathbf{X})\right]}{n}$$

Thus, the probabilistic error bounds of Monte Carlo methods scale at $O(n^{-1/2})$ independently of the input dimension D and the smoothness of the function. This is clearly an advantage over other numerical integration methods. For example, the trapezoidal rule assumes that the integrand is twice differentiable and uses linear interpolation. Since more structure of the function is used its error is $O(n^{-2/D})$, which is better than for Monte Carlo only when $D < 4$.

3.3.2 Stratified sampling and Quasi Monte Carlo

In the simplest case, independent samples from $p(\mathbf{x})$ are used but better convergence rate can be obtained with stratified sampling and Quasi Monte Carlo, at least for functions f with some spatial regularity. These methods ensure that samples are distributed more or less uniformly over the domain. Stratified sampling consists in breaking up the domain of integration $\mathcal{A}$, into m disjoint subdomains $\mathcal{A}_i$, then defining:

$$I_i = \int_{\mathcal{A}_i} f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \qquad \text{with } \mathbf{x} \in \mathcal{A}_i \tag{3.17}$$

Then all the I_i are estimated separately using Monte Carlo methods. The integral I is computed by summing the estimates I_i. The subdomains $\mathcal{A}_i$ are generally chosen in a way that their sizes are related to the *pdf* $p(x)$.

Remember that the variance of the classical Monte Carlo estimator depends on the variance of the integrand. By breaking up the integrand into several parts, over the subdomains $\mathcal{A}_i$, each part of the integrand generally has a lower variance than the full integrand. Hence each estimate $\hat{I}_i$ generally have a lower variance than $\hat{I}_p$. In general, stratification will never increase variance, though in some situations it will not reduce variance either. Stratified sampling is very effective for low-dimensional integration problems where the integrand is reasonably well-behaved. However, if the dimension is high or if the integrand has singularities or rapid oscillations in value, then stratified sampling will not help significantly [Vea98, Mit96].

On the other hand, Quasi Monte Carlo methods try to distribute the samples as uniformly as possible by choosing their location deterministically. There is no more randomness in the positions of the samples, and when applying Quasi Monte Carlo to computer graphics, special care must be taken in order to avoid aliasing and/or artifacts. The goal of Quasi Monte Carlo methods is generally to minimize the irregularity of distribution, for example by minimizing the star discrepancy of the samples set. To achieve this goal, they use sophisticated space filling designs such as the Latin Hypercube design [MBC00], Sobol Lattices [Sob93] or the MaxiMin and MiniMax criteria [JMY90]. Several Quasi Monte Carlo methods allow to draw additional samples, while keeping a minimal discrepancy, which is not easy to do with basic stratification.

3.3.3 Importance sampling

If sampling directly from $p(\mathbf{x})$ is difficult, or the sampling density of $p(\mathbf{x})$ do not match the regions where the value of $f(\mathbf{x})$ is high, it may be preferable to draw samples from a more suitable distribution $q(\mathbf{x})$. For this, equation (5.1) is rewritten as:

$$I = \int \frac{f(\mathbf{x})p(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) d\mathbf{x}$$

to obtain the estimate:

$$\hat{I}_q = \frac{1}{n} \sum_{i=1}^{n} \frac{f(\mathbf{X}_i)p(\mathbf{X}_i)}{q(\mathbf{X}_i)}$$

$\hat{I}_q$ is also an unbiased estimate of I but its variance becomes:

$$\mathrm{Var}(\hat{I}_q) = \frac{1}{n}\mathrm{Var}\left(\frac{f(\mathbf{X})p(\mathbf{X})}{q(\mathbf{X})}\right) = \frac{1}{n}\left[\int \frac{f(\mathbf{x})^2 p(\mathbf{x})^2}{q(\mathbf{x})} d\mathbf{x} - I^2\right]$$

If $f(\mathbf{x})$ is always non-negative, the variance of this estimator is zero when $q(\mathbf{x})$ differs from $f(\mathbf{x})p(\mathbf{x})$ by a constant scale factor [OZ00]. This suggests that a good importance sampling density $q(\mathbf{x})$ should be roughly proportional to $f(\mathbf{x})p(\mathbf{x})$. However, as shown in [OZ00], this strategy can fail dramatically when $q(\mathbf{x})$ decreases towards zero faster than $f(\mathbf{x})^2 p(\mathbf{x})^2$ and it may even results in $\mathrm{Var}(\hat{I}_q) = \infty$. Moreover, if the importance sampling density $q(\mathbf{x})$ is zero in regions where $f(\mathbf{x})p(\mathbf{x})$ is not zero, these regions will never be explored and the estimator will be biased. Both problems can be solved by using a weighted sum of sampling densities (mixture sampling) [OZ00]. Importance sampling is one of the most used variance reduction technique in global illumination.

3.3.4 Control variates

A good presentation of this variance reduction method can be found in [HLO04]. Here we simply summarize some important results. For the sake of clarity, we will assume that $p(\mathbf{x}) = 1$ for all the developments in this section, which means that the samples $\mathbf{X}_i$ are drawn from a uniform distribution. The underlying idea in control variates is to exploit a crude approximation $g(\mathbf{x})$ of the integrand $f(\mathbf{x})$ and its integral to sharpen the estimate of I. Let $\mathbf{g}$ be a vector of functions $(g_1, \ldots, g_J)^t$ and $\mathbf{G} = (G_1, \ldots, G_J)^t$ be the vector of integral values:

$$\mathbf{G} = \int \mathbf{g}(\mathbf{x}) d\mathbf{x}$$

Then for any vectors $\beta = (\beta_1, \ldots, \beta_J)^t \in \mathbb{R}^J$ the value of $\hat{I}_\beta$:

$$\hat{I}_\beta = \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathbf{X}_i) - \sum_{j=1}^{J} \beta_j \left(g_j(\mathbf{X}_i) - G_j \right) \right) = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{X}_i) - \beta^t \left(\mathbf{g}(\mathbf{X}_i) - \mathbf{G} \right) \tag{3.18}$$

is an unbiased estimate of I. Its variance:

$$\mathrm{Var}(\hat{I}_\beta) = \frac{1}{n} \mathrm{E}\left[\left(f(\mathbf{X}) - I - \beta^t \left(\mathbf{g}(\mathbf{X}) - \mathbf{G} \right) \right)^2 \right]$$

is a quadratic function of the vector β. Its minimum value is obtained for the vector $\beta = \beta_{opt}$ given by:

$$\beta_{opt} = \left(\int \left(\mathbf{g}(\mathbf{x}) - \mathbf{G} \right) \left(\mathbf{g}(\mathbf{x}) - \mathbf{G} \right)^t d\mathbf{x} \right)^{-1} \int \left(\mathbf{g}(\mathbf{x}) - \mathbf{G} \right) f(\mathbf{x}) d\mathbf{x}$$

As in most cases this expression cannot be computed analytically, β_{opt} is estimated by:

$$\hat{\beta}_{opt} = \left(\sum_{i=1}^{n} \left(\mathbf{g}(\mathbf{X}_i) - \hat{\mathbf{G}} \right) \left(\mathbf{g}(\mathbf{X}_i) - \hat{\mathbf{G}} \right)^t \right)^{-1} \sum_{i=1}^{n} \left(\mathbf{g}(\mathbf{X}_i) - \hat{\mathbf{G}} \right) f(\mathbf{X}_i) \tag{3.19}$$

where $\hat{\mathbf{G}} = (\hat{G}_1, \ldots, \hat{G}_J)^t$ and:

$$\hat{G}_j = \frac{1}{n} \sum_{i=1}^{n} g_j(\mathbf{X}_i)$$

Let us note that $\hat{\beta}_{opt}$ is the vector of regression coefficients for the ordinary least squares estimator relating $f(\mathbf{X}_i)$ to the $g_j(\mathbf{X}_i)$ values, that is why this method sometimes goes by the name of "regression estimators" in the literature.

The optimal control variate estimator is then obtained by substituting $\hat{\beta}_{opt}$ for β in Equation 3.18. If the same samples are used for computing $\hat{\beta}_{opt}$ and the estimate $\hat{I}_{\beta_{opt}}$, a small bias is introduced but it can be neglected in practical [HLO04].

Both importance sampling and control variate methods use an approximating function of the integrand but as this function is used as a probability density in the case of importance sampling, its choice is more constrained as discussed in section 3.3.3. The above theoretical developments can be easily extended to the case where importance sampling (with or without mixture sampling) is used concurrently with control variates (see [OZ00]). However, the case of mixture sampling leads to a singular regression which can be solved using a singular value decomposition (SVD) method for computing the vector $\hat{\beta}_{opt}$ [OZ00].

3.3.5 Adaptive sampling

The idea of adaptive sampling is to take more samples where needed. In order to evaluate the number and the location of these additional samples, adaptive sampling takes into account the samples that have already been taken. Typically, adaptive sampling computes the variance of a subset of the obtained samples, and then decides to take more samples if the variance exceeds a given threshold. Several techniques using adaptive sampling have been proposed in computer graphics [Mit87, PS89, TJ97]. The main difference with a strategy like importance sampling is that the decision to add more samples is taken "on the fly" rather than using an a-priori knowledge.

3.3.6 Metropolis light transport

Metropolis Light Transport [VG97] uses a variant of the Monte Carlo method to solve the rendering equation. The algorithm constructs paths from the eye to a light source, and then constructs slight modifications (mutations) of the paths. Each mutation is accepted or rejected with a carefully chosen probability, to ensure that the paths are sampled according to the contribution they make to the image. Metropolis light transport performs especially well on problems that are usually considered difficult, and can be orders of magnitude more efficient than the previous Monte Carlo approaches. Like adaptive sampling, Metropolis light transport relies on a knowledge which is built from the already computed samples, during the rendering.

3.4 Our approach to global illumination

In this thesis, we first focus on combining photon mapping and irradiance cache to generate a view independent global illumination solution. Our algorithm called Photon-Driven Irradiance Cache allows fast preview of the photon map solution, as well as a refinement step which yields high quality results. The global illumination solution is then interactively displayed during a walkthrough. Secondly, we investigate the first application of a variance reduction method used in statistics, the Bayesian Monte Carlo, to the domain of global illumination. We show that Bayesian Monte Carlo is able to exploit a prior probabilistic hypothesis on the radiance in the scene to achieve a better rendering quality. Since classical Bayesian Monte Carlo is very costly, we propose a implementation to make Bayesian Monte Carlo practical for global illumination.

Photon-Driven Irradiance Cache Photon mapping is the method of choice for rendering complex lighting effects, taking into account multiples bounces at a low cost. However, fast visualization of the photon map is usually very noisy. In addition, to achieve high quality rendering a costly final pass called final gathering has to be performed. Since final gathering is view dependent, it has to be performed each time the user move the camera.

We propose to use the photon map to compute a view independent solution covering the whole scene. To improve the quality of a fast preview of the photon map, we build on the idea proposed by irradiance caching to reduce the number of queries to the photon map, while yielding visually pleasant results. In addition, by creating irradiance records covering the whole

scene, only few extra computations are needed during the rendering step. We also propose to reduce the number of rays cast during the final gathering pass by reusing the visibility information contained in the photon map. Finally, the computed global illumination solution can be interactively displayed during a walkthrough.

Bayesian Monte Carlo for global illumination Most of the methods used in global illumination makes extensive use of Monte Carlo method. Photon-Driven Irradiance Cache is no exception: Monte Carlo integration is used by the photon mapping algorithm to estimate irradiance from the photon density, as well as the final gathering used to compute the final irradiance value associated to each irradiance record. One of the major problems of Monte Carlo methods is the inherent variance associated with the estimator, which results in noise in the rendered pictures. While several existing variance reduction techniques have been successfully applied to global illumination, to our knowledge it is the first time that Bayesian Monte Carlo is investigated in this domain.

We show that Bayesian Monte Carlo can be used to produce better quality rendering, when computing diffuse interreflections. As Bayesian Monte Carlo exploits a prior probabilistic model to reduce the variance of the Monte Carlo estimator, we had to investigate how it would apply to the global illumination problem. We show that even when using suboptimal hyperparameters in our model, Bayesian Monte Carlo yields a better estimate with less variance, using the same number of samples. However, the cost of the classical Bayesian Monte Carlo is prohibitive in the case of global illumination. We propose an implementation which solves this problem, by computing quadrature coefficients during a precomputation step. At rendering step, the values of each sample are multiplied by the adequate quadrature coefficients, yielding the estimate value. Therefore, Bayesian Monte Carlo yields better renderings than classical Monte Carlo, for the same amount of time. We noticed that the improvement brought by Bayesian Monte Carlo, when using globally optimized hyperparameters, depends on the sampling method. Particularly, when compared to stratified importance sampling, it is difficult to notice the difference. To improve the rendering quality, we propose a new way of determining the sample locations, based on a minimization of the a-priori posterior variance of the estimate. This a-priori "optimal" sampling, used with the Bayesian Monte Carlo estimator, yields same quality rendering as that provided by stratified importance sampling, while using less samples.

Chapter 4

Photon-Driven Irradiance Cache

4.1 Introduction

Today, two of the most commonly used global illumination computation methods are photon mapping [Jen96] and irradiance caching [WRC88]. Based on Monte Carlo ray tracing, both methods have their own advantages and drawbacks. Photon mapping is view-independent but needs a costly second pass to achieve high quality rendering, called *final gathering*. This pass is computationally expensive, which is a serious problem when the aim is interactive rendering of globally illuminated scenes. As for irradiance caching, it is a faster method but the calculation it involves is view dependent. To make this method view-independent, one has to run it for a high number of cameras to get records (cached information) over the entire scene.

Radiosity [GTGB84] can also be used to compute diffuse inter-reflection. This method exhibits discontinuities due to meshing and thereby needs postprocessing. It is also time consuming. Instant radiosity [Kel97] is also a Monte Carlo based method. It generates virtual point lights (VPL) that are used to compute the diffuse inter-reflection components within a scene. Indirect glossy reflection seems difficult to evaluate using the VPLs. Laine *et al.*[LSK+07] propose an incremental instant radiosity that consists in reusing the VPLs and incrementally maintaining their distribution over the scene. This method considers only one light bounce to achieve interactivity. [LZT+08] describes a global illumination method based on point sampling. Likewise, our approach make use of points (records) to cache the global illumination solution. For the two approaches, point placement is performed in a similar way. First, a set of candidate points is determined by particle tracing, then only a subset of these candidates is kept, based on some heuristics. We use the reflection properties of the materials to guide the particle tracing as well as a criterion based on harmonic mean distance [WRC88], while [LZT+08] makes use of importance as well as another criterion relying on Poisson disk distribution with a new distance function. However, global illumination is determined differently with the two approaches.

The goal of Photon-driven Irradiance Cache is to propose a global illumination method that: (1) does not need any geometry meshing, (2) computes indirect diffuse illumination in densely occluded scenes by handling multiple light bounces, (3) allows fast acceptable preview of the computed global illumination solution, (4) renders this solution in real-time, (5) exploits spatial

coherence, (6) can be part of a global illumination algorithm accounting for diffuse, specular and glossy materials, for which the indirect diffuse component (as a result of L(D|G)*DE paths, where G corresponds to moderate glossy reflections) is computed by our method while the glossy and specular inter-reflections are evaluated by using classical methods.

To achieve this goal, our method takes advantage of the photon mapping and irradiance caching techniques and combines them for fast and accurate global illumination computation. Starting with a photon map, an irradiance cache is created based on the photon positions. The irradiance assigned to each record having been computed coarsely using the nearby photons, we perform an additional refinement step for high quality rendering.

This chapter is organized as follows. We first present our motivations in Section 4.2. Then Section 4.3 details our approach of global illumination for interactive walkthrough. We also discuss the use of control density estimation in order to reduce the computation time of our method. Finally, Section 4.6 presents some results.

4.2 Motivations

Photon mapping [Jen96] allows to simulate any complex lighting by tracing particles through the scene. The irradiance incoming at visible point is estimated by computing the density of the particles at a given location. Note that the lighting simulation pass is view-independent. The photon mapping technique provides multiple-bounce global illumination. In addition, the direct use of the information contained in the photon map yields noisy pictures. However to obtain a high quality rendering, a costly pass of *final gathering* must be performed by tracing rays from each visible point. The computation time of the final gathering is prohibitive, especially in the context of interactive applications, since it has to be performed for each different viewpoint.

Irradiance caching [WRC88] algorithm exploits the spatial coherence of the indirect lighting. The irradiance is sparsely sampled at some points in the scene and stored into *irradiance records*, then interpolated for every other point. The creation of the records is defined by the current viewpoint: the algorithm computes all the records needed to interpolate irradiance at each visible point. However, the irradiance value computed at a record position is independent of the viewpoint. Therefore, in a static scene, images from other viewpoints can be generated quickly by reusing the previously created records, instead of recomputing a novel global illumination from scratch. In addition, the created irradiance cache can be rendered interactively using radiance cache splatting [GKBP05].

Radiance cache splatting [GKBP05] uses the GPU, to compute irradiance and radiance records, and considers only one-bounce reflection. Rendering is performed by splatting the records on the image plane and runs at interactive frame rates. Note that the splatting process can be used to display any cache, including those taking into account multiple bounces of light. When the camera moves, additional records have to be computed, which slows down rendering. As radiance cache splatting is view dependent (the creation of records is defined by the current viewpoint) the user has to manually place virtual cameras so that the entire scene is covered by irradiance records. This is tedious and computationally expensive. However, once these records have been computed, rendering can be performed in real-time. Still, this

method cannot easily be used in the context of interactive applications due to the necessary computation of new irradiance records for each frame.

Note that one could use a photon map and final gathering to compute irradiance records for a given viewpoint. However, this naive approach does not solve the view dependency problem (creation of new records when the camera moves). In addition, it is not possible to obtain a fast preview of the global illumination solution.

Our aim is to combine the generality of the photon mapping algorithm and the computational efficiency of irradiance caching for interactive walkthrough in global illuminated scene. To achieve this goal, our algorithm computes an irradiance cache directly from the information contained in the photon map. The cache accounts for multiple-bounce reflections and covers most parts of the scene without any user intervention.

First, we propose a fast method for generating coarse irradiance cache records using only the information contained in the photon map, without casting any new rays. These records cover the whole scene without having to define many potential viewpoints. Once the cache is filled, the radiance cache splatting algorithm can be used to render the global illumination solution interactively. During a following pass, the coarse irradiance cache is refined using a modified final gathering pass, which use information from photon map to reduce the number of rays cast.

4.3 From photon map to irradiance cache

Our algorithm can be split into three parts. First, we create the photon map. Second, starting with an empty irradiance cache, record positions as well as a coarse approximation of irradiance at these records are computed from the photon map. The resulting cache can be displayed using radiance cache splatting (or converted into light maps). In the third pass, the irradiance assigned to each record is refined and irradiance gradients are computed.

Finally we discuss density control for the photon map and its benefits for scenes with complex geometry and/or illumination.

4.3.1 First pass: photon map construction

We use standard photon mapping [Jen96]. Our photons store additional data that will be computed and used during the second and third passes of the algorithm:

- cumulative weight contribution of the surrounding records at the photon location: w_{sum},
- cumulative indirect irradiance contribution due to surrounding records,
- irradiance due to direct lighting (precomputed before refinement).

4.3.2 Second pass: irradiance cache from photon map

As photons cover all surfaces which are indirectly lit, their positions are used as potential record positions. Starting with an empty cache we add progressively new records by examining each photon in the map. This second pass is described by Algorithm 2.

To decide whether to create a new record at a given photon position p, we find the set S of the already computed records surrounding p (Equation 3.12). If S is empty or if the cumulative

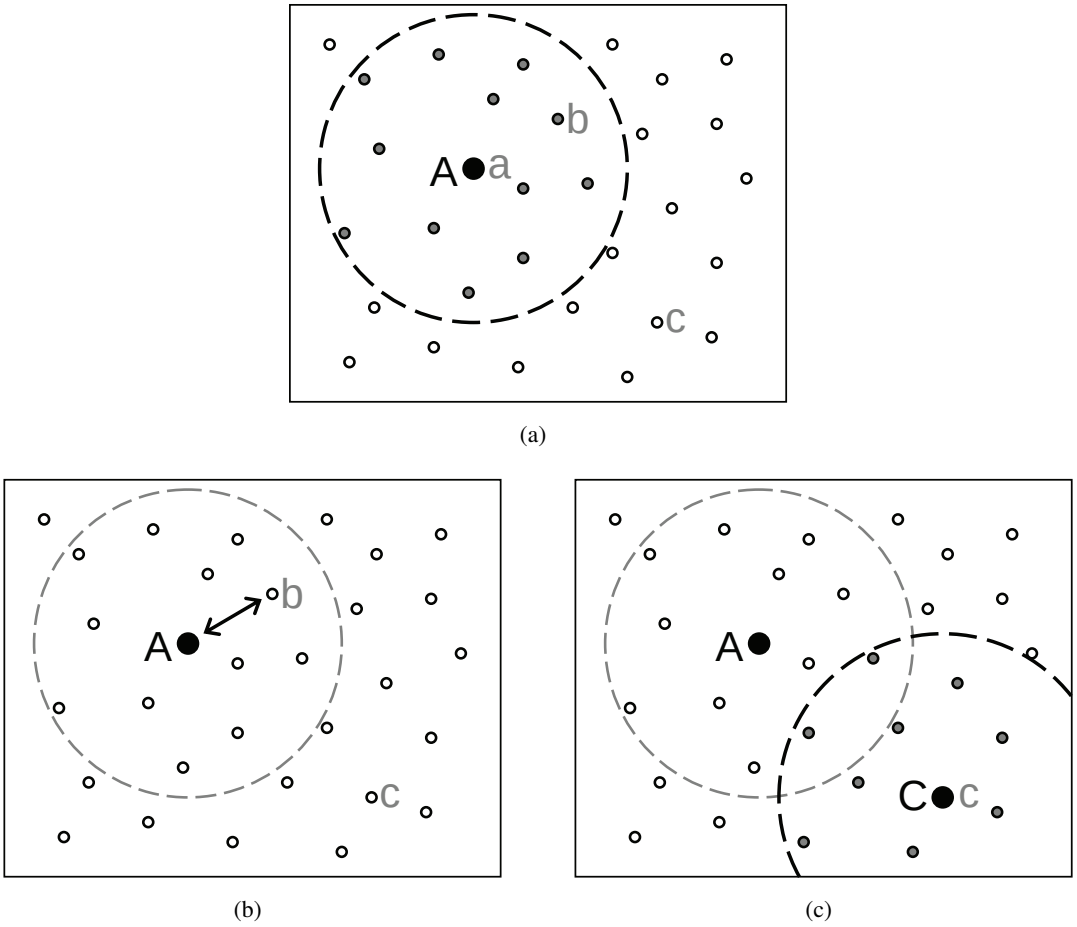


Figure 4.1: We start with an empty irradiance cache, and process photons a, b and c. *(a)* As no records are present around photon a, a new record A is created at photon location. *(b)* As photon b is close to record A, $w_{sum} > a$, hence we do not create a new record. *(c)* Photon c is out of the zone of influence of A. A new record C is created at its location.

weight, $w_{sum} = \sum_{k \in S} w_k(p)$, is less than a, we create a new record at position p. Because the number of photons in the photon map is usually high, building S for each photon then estimating the cumulative weight would be very expensive. Rather, for each photon we store the cumulative weight contribution of the already computed records surrounding the photon, w_{sum}. Since we start with an empty cache, w_{sum} is initialized to zero for every photon, then updated for each creation of a new record. In a similar way, each photon stores the irradiance at its position, derived from the existing irradiance cache. This value is also initialized to zero, then updated. When processing a photon, if $w_{sum} < a$, we create a new record k at the photon position and compute its associated E_k and R_k (Figure 4.1) from the nearby photons.

In photon mapping, irradiance at a given point is related to the density of photons around this point [Jen96]. Density estimation [Sil86] requires two parameters, kernel and bandwidth. The Epanechnikov kernel minimizes the error of density estimation, and is easy to compute.

Algorithm 2: Generation of irradiance records

```
Input: the photon map PM
Output: an irradiance cache covering the scene
/* GenerateIrradianceCacheFrom(PM)                                   */
begin
    foreach photon p ∈ PM with w_sum < 1/a do
        find S_n, set of the n-nearest photons;
        estimate R_k from S_n;
        find S_{a.R_k}, set of all photons within a sphere of radius a.R_k around p;
        estimate E_k from S_{a.R_k};
        create record k at photon location;
        foreach photon of S_{a.R_k} do
            update w_sum;
            update cumulative indirect irradiance;
        endfch
    endfch
end
```

The choice of the bandwidth is more complex. A low bandwidth results in a noisy irradiance reconstruction, whereas large bandwidth in a smooth but biased estimation.

To obtain a more visually pleasant approximation, we rather choose a larger bandwidth than in standard photon mapping. Using the observations of [WRC88], we choose to set the bandwidth to the radius of the influence sphere, $a.R_k$. E_k is then estimated using only the photons lying in the influence sphere of record k.

The radius of the sphere of influence depending on R_k, we need to compute the harmonic mean distance to objects visible from p_k. This is usually performed by sampling the hemisphere around p_k and casting rays through the scene (Figure 4.2(a)). To avoid ray casting during this pass, we estimate R_k using the neighbor photons. During the creation of the photon map, each photon is assigned a direction of incidence and the distance from the last hit point. We then compute the distance between this last hit point and p_k. Using the n-nearest neighbors, we compute an estimate of R_k (Figure 4.2(b)).

4.3.2.1 Neighbor clamping

Depending on a record k and on the number of photons in its neighborhood, we may miss some objects when estimating R_k (Figure 4.2(c)), due to undersampling. If the computed R_k is far larger than it should be, the corresponding record will be assigned a badly sized zone of influence which will cause interpolation artifact during the rendering. However, the impact of a bad estimation of R_k is dramatically reduced by using the *neighbor clamping* method [KBPZ06]. Neighbor clamping allows a given record k to propagate its estimated value of R_k to neighbor records k_i, in order to put an upper bound on their R_{k_i} values. Let us take two records A and B. Due to a small number of photons lying around B, R_B is overestimated and do not take into account the presence of a neighbor sphere (Figure 4.3(a)). Record A has been

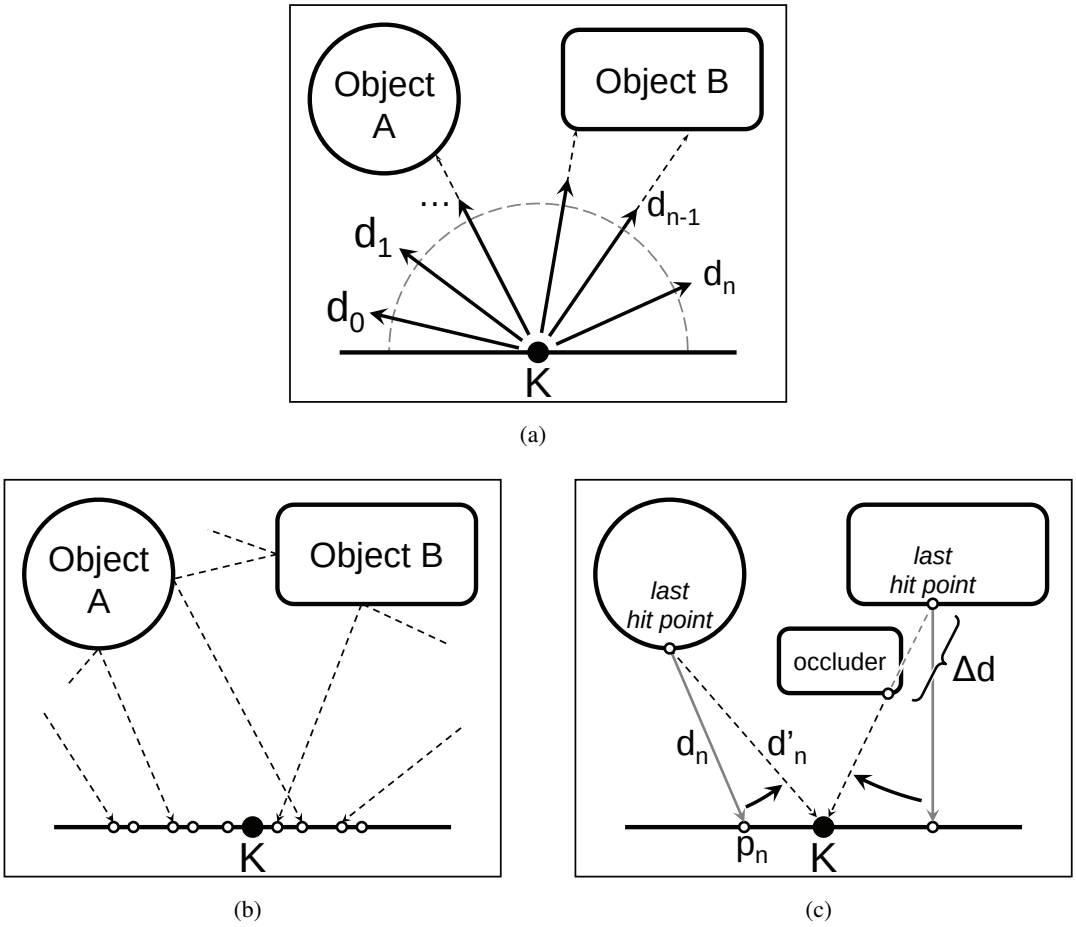
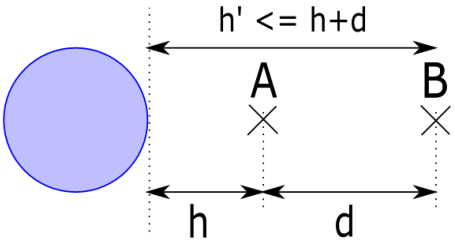


Figure 4.2: Estimation of R_k. *(a)* Usually R_k is computed by casting N rays towards the scene: $R_k = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{d_i}$. *(b)* In our method, we use the information contained in the photons surrounding record K. *(c)* Each photon p_n stores the distance d_n to the last hit and its incident direction. From these two data we deduce d'_n, distance between K and the last hit of the photons. Although we use photons that are close to K, we may miss some occlusions, and overestimate d_n.

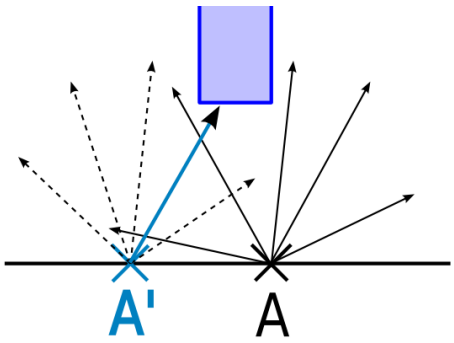
assigned a correct value R_A. Since the distance between record A and B is d, we clamp the value of R_B to $R_A + d$. This process allows us to reduce number of the problematic cases where harmonic mean distances are badly estimated: the photons lying in the zone of influence of a record did not already bounce of a close wall or a neighbor object (Figure 4.3(b) and 4.3(b)).

4.3.2.2 Density estimation and boundary bias

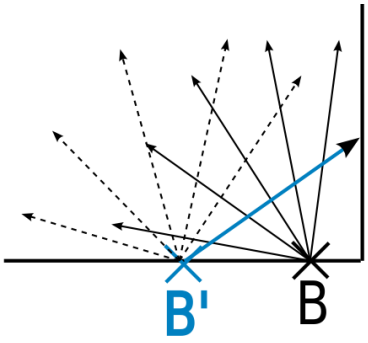
When rendering from a photon map, irradiance is estimated for each pixel of the image. Hence, the noise is distributed over the pixels of the image. If E_k is computed in the same way, the noise will be distributed over the irradiance records. As a record can have influence over



(a) Neighbor clamping principle



(b) Record A missed a close object.



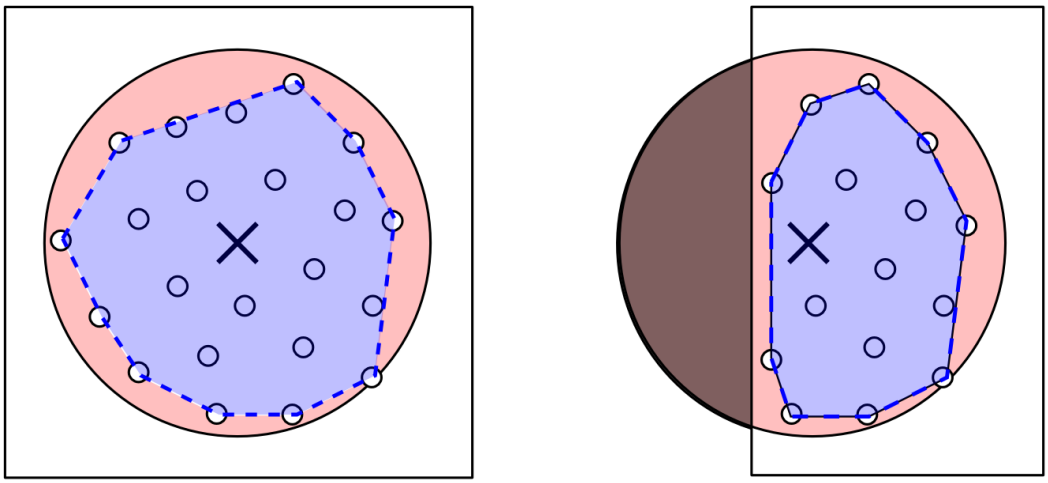
(c) Record B missed a close wall.

Figure 4.3: Impact of a bad estimation of R_k is dramatically reduced by using neighbor clamping. *(a)* and *(b)* In each case, the records A' and B' are able to propagate a reasonnable upper bound on the harmonic mean distances of records A and B.

hundreds of pixels of the final rendering, noisy records will result in unpleasant visual artifacts. Recall that during the second pass we only compute an approximation of the irradiance which will be refined later.

Each record creation requires two queries to the photon map: a n-nearest neighbors search used to estimate R_k, and a query to find the set $S_{a.R_k}$ of all photons within the range of $a.R_k$ around record position. Once E_k and R_k have been computed, the new record k is added to the cache. Photons lying within the influence sphere of k get their cumulative weight updated with the new contribution of k. Then, the following photon of the photon map is processed. When all the photons have been processed, the irradiance cache is filled with records covering most of the scene. This coarse irradiance cache can already be used with radiance cache splatting for real-time rendering.

As explained before, classical density estimation from the photon map is subject to inaccuracies. In the absence of any surface boundary, the area supporting the photons is usually computed by taking the area of the disc whose radius is given by the distance to the farthest photon (Figure 4.4(a)). In the presence of a surface boundary, the area supporting the photons should be evaluated as the surface of the previously defined disc minus the surface of circular segment which is lying outside the surface (Figure 4.4(b)). This second case is often neglected, which leads to the rendering problem called *boundary bias*.



(a) (b)

Figure 4.4: Classically, the area sustained by the photons is computed by taking the intersection between the search sphere and a planar surface. Due to the relatively low number of density estimations our algorithm performs, we are able to compute this area by using a convex hull approach. (a) shows a case where there is no boundary. (b) shows a case where there is a boundary. Typically, the area would be overestimated, leading to blackened boundaries during the rendering. The convex hull approach allows us to compute a more precise area.

This problem could be solved using a convex hull containing the photons, represented by a polygon, as explained hereafter. The surface of the polygon corresponds to the area supporting the photons. However in classical rendering from the photon map, the number of density estimations performed can be higher than several hundreds of thousands, then computing all the convex hulls is very time consuming. Unlike classical photon mapping, our algorithm performs density estimation only for new records, with a maximum of two density estimations performed per created record. As the creation of the cache from the photon map is fast, and it is not the most time consuming part of the whole algorithm, we can afford the computation of the convex hulls. Consequently, when computing the density of photons around a given point, the sustained area ΔA is not approximated anymore by projecting the search sphere on a planar surface. Rather, this area is determined by computing the convex hull of each photon set (Figure 4.4).

However, when using a convex hull to compute the area supporting the photons, another problem arises. It can be easily shown that the convex hull area estimation yields a value which is always smaller than the true area (Figure 4.4). The difference between the area computed using a convex hull and that of the disc, or the partial disc, depends on the number of photons. When the number of photons n tends to infinity, the convex hull area converges to the exact area. However, for a small number of photons, the area estimated using convex hull estimation can be far smaller than the exact area. This results in a boundary bias around the edges of the surface geometry: as the area supporting the photons is underestimated, the density estimated close to

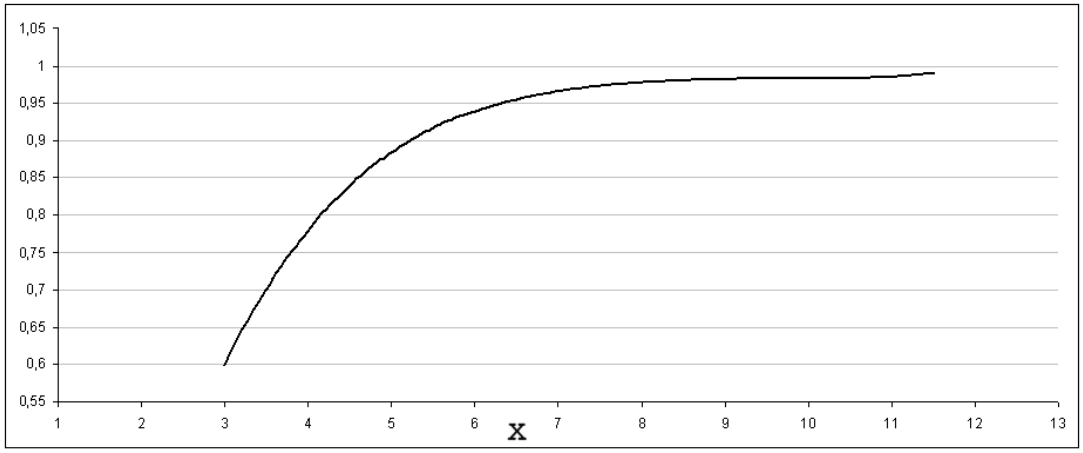


Figure 4.5: Mean ratio between the area computed using a convex hull area and the area subtended by the perfect disc. The former one is always smaller than the second one, and the difference is depending on the number of photons present in the set. x is the natural logarithm of the number of photons. When the number of photons is larger than 1000 ($x > 7$), ratio is larger than 0.95.

surface boundaries is overestimated. In rendered images, we observe that the boundaries get lighter than the correct result. This is because photon queries around the boundaries generally yield a lower number of photons. To overcome this problem, we propose to divide the area of the convex hull by a coefficient $P(x)$, lower than 1 and depending on the number of the photons within this area. Experiments have shown that the following function is a good compromise:

$$\begin{aligned} P(x) &= 3.519 \times 10^{-6} x^{-6} - 1.375 \times 10^{-4} x^{-5} \\ &+ 1.936 \times 10^{-3} x^{-4} - 9.222 \times 10^{-3} x^{-3} \\ &- 3.791 \times 10^{-2} x^{-2} + 5.426 \times 10^{-1} x - 5.636 \times 10^{-1} \end{aligned}$$

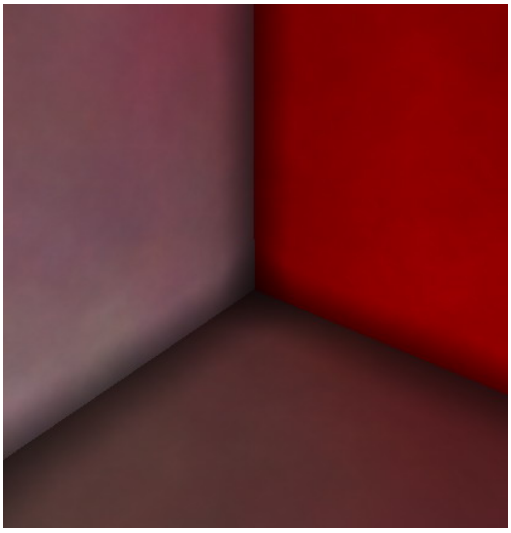
where $x = ln(n)$ and n is the number of photons contained in the set.

Let us see how we determine this function. We consider a disc truncated by a straight border. We generate then several millions of distributions of photons within this truncated disc. Then for each fixed number of photons and each distribution we compute the ratio of the convex hull area to the exact area. After fitting we get function $P(x)$ (Figure 4.5).

The figure 4.6 shows the comparison between rendering from the photon map and from the irradiance cache generated during the second pass of our method, with and without convex area hull estimation.

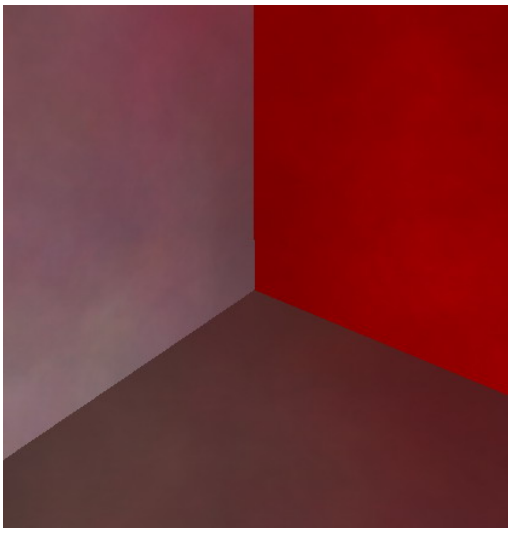
4.3.3 Third pass: refining the cache

The irradiance cache computed during the second pass stores only a coarse approximation of irradiance. In this third pass we compute a more accurate value of E_k as well as irradiance gradients for each record. We perform this refinement using a slightly modified final gathering

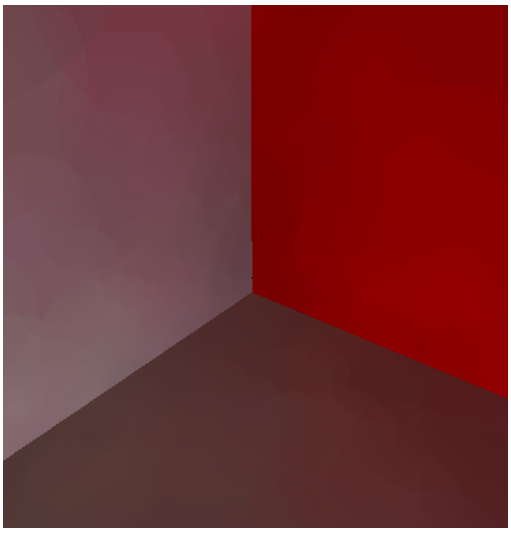


(a) Photon mapping, 15.2 sec

(b) Irradiance cache, 1.44 sec



(c) Photon mapping, with CHAE, 58.5 sec



(d) Irradiance cache, with CHAE, 2.55 sec

Figure 4.6: Elimination of boundary bias when generating the irradiance cache. *(a)* and *(b)* shows renderings using classical density estimation. A strong boundary bias is visible. *(c)* and *(d)* Using convex hull area estimation (CHAE) eliminate the boundary bias. However, the rendering time when performing a density estimation for each pixel is prohibitive. Our method greatly reduces the number of density estimations performed and does not forbid the use of convex hull area estimation.

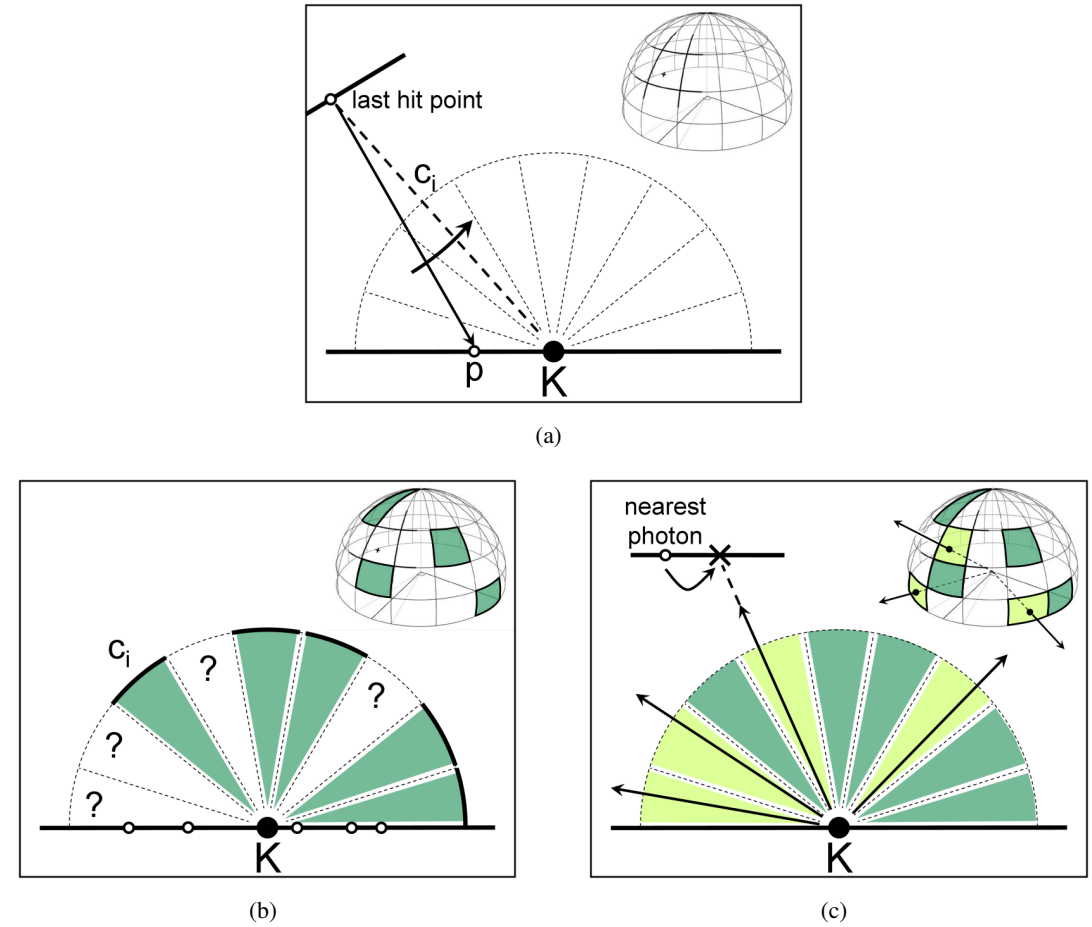
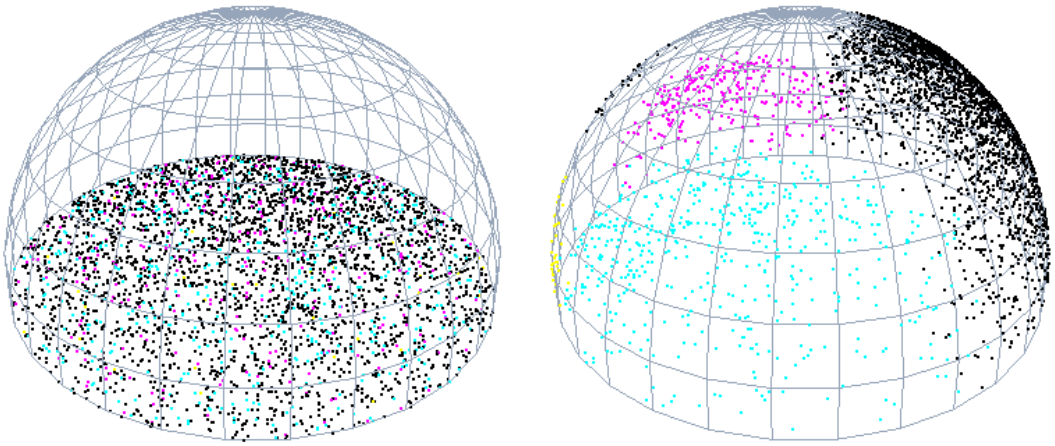


Figure 4.7: Refinement of a record K. *(a)* p is a neighbor photon. Its associated hit point is reprojected onto the hemisphere above K. This reprojection yields a *virtual ray* which intersects the cell c_i. *(b)* After processing all the photons, the hemisphere above K is partially filled with *virtual rays*. *(c)* We shoot rays only for the cells which have not already been considered by any reprojection. The contribution of each ray is given by the photon closest to the hit point.

that exploits information given by the nearest photons. This information allows reducing the number of rays traced from the considered record. Note that this refinement can be performed while the user is walking through the scene interactively, the scene being rendered using the current irradiance cache. In our implementation, the visible records are refined in priority.

In the classical final gathering approach the hemisphere above the record location is sampled by tracing a high number of rays. In our approach we aim at reducing the number of rays by reusing the path of the photons lying within the zone of influence of the record. More precisely (Figure 4.7), for each photon lying in the zone of influence of a record k, we first retrieve the ray incident to this photon (each photon stores a direction of incidence and the distance to the last hit point). Assuming that the visibility at the location of the photon and the location p_k of the record are not significantly different, we reproject the last hit point p_{lh} of the photon to



(a) Photons lying inside the zone of influence of a record (b) The same photons reprojected over the hemisphere

Figure 4.8: Example of reprojection of photons inside the zone of influence of a record over the hemisphere.

create a *virtual ray* between p_k and p_{lh}.

Let us now consider a hemisphere H_k divided into cells, located above p_k (Figure 4.7(a)). We determine the cell c_i intersected by the *virtual ray* (Figure 4.7(a)) and we assign the *virtual ray* to this cell as if an actual ray had been traced. Once all the photons within the zone of influence of record k have been processed, the hemisphere H_k is partially filled with *virtual rays* (Figure 4.7(b)). The remaining cells are filled by actually casting rays as in the classical final gathering approach (Figure 4.7(c)). Note that we assumed a negligible change of visibility across the zone of influence of a record, which may lead to errors in the computation. [BDT99] re-uses *primary* rays using radiance interpolants and error bounding, while our approach re-uses *secondary* rays during final gathering. It turns out that our naive reprojection does not entail visible artifacts. However, radiance interpolants [BDT99] could be used during the third pass. Figure 4.8 shows the reprojection of a set of photons.

Each cell c_i is assigned a ray and a hit point. As in [Chr99], the irradiance stored in the nearest photon data structure is assigned to this hit point. In our implementation, each photon stores a reference to the previous photon along its path. In the case of a *virtual ray*, the nearest photon to the hit point can then be retrieved immediately. The irradiance of the nearest photon is then multiplied by the diffuse reflectance at the hit point to get the radiance incoming through the cell. The radiances calculated for all the cells are used to compute the refined irradiance value as well as the irradiance gradients of the record using [KGBP05, WH92].

4.4 Density controlled photon map

When building a standard photon map in the first pass, two problems may arise in the second and third passes of our method: missing irradiance records on some surfaces and an insufficient number of photons for the refinement of some records. These two problems are

related to the distribution of the photons in the scene. Let us first recall that, in the second pass, records are placed at photon positions. If the photon density is low on a surface while R_k is not large enough, the zones of influence of the created records may not overlap. Consequently, this gives rise to holes corresponding to non covered zones for which new records have to be computed at the rendering step (Figure 4.9(a)). Secondly, during the third pass, the speed-up achieved during the refinement of the records is related to the number n of photons lying in the zone of influence of each record. This number depends on the number of cells used for the final gathering: with too few photons we have to cast a lot of new rays, while with too many photons a significant number of rays be reprojected onto the same cells, wasting then computation time (Figure 4.9(b)). We would like to have a reasonably constant number of photons inside the zone of influence of a record.

One solution to this problem is to increase the number of photons in the photon map, but a larger photon map results in a larger computation time for the first and second pass. An ideal solution would lead to a situation in which the zone of influence of each record, whatever its size, contains approximatively the same number of photons. Suykens *et al.* [SW00] proposed a method to control the density of photons stored in a photon map. This can be done either during the construction of the photon map (by not storing some of the emitted photons) or during a second pass (by deleting a part of the already stored photons). The power carried by the discarded photons is then redistributed over the neighboring photons. We propose a new photon density function which allows to store more photons in the parts of the scene where R_k is low (such as corners or surfaces with complex visibility), and few photons in regions with large R_k. This results in shorter computation time for the second and third pass. We discuss the viability of this approach by comparing this speed-up with the overhead associated with the density control pass.

4.4.1 A view-independent density control function

In standard photon mapping, density control [SW00] is usually used in combination with importance sampling. Density control reduces the number of photons lying in unimportant regions (i.e. regions not contributing to the image). As the targeted density control function is based on importance, the obtained photon map is then view-dependent. Unlike Suykens *et al.*, we need a view-independent photon map to create an irradiance cache. Therefore, our new density control function does not depend on importance but on the harmonic mean distance from a point to visible objects in the scene. Indeed, as explained above, the speed-up achieved during the refinement of the records is related to the number n of photons lying in the zone of influence of each record. We would like this number to be close to the number of cells used for the final gathering. As the zone of influence of each record depends on R_k (i.e. the harmonic mean distance to other visible objects in the scene), our density control strategy depends on two variables: n and R_k.

Given a control density function, two methods exist to create a photon map. For the first one [SW00] the actual and targeted densities at each hit point are compared during the course of the creation of photon map, which allows to decide whether to store or not a photon. The photons have to be stored in an efficient data structure (kd-tree) which has to be updated whenever a photon has to be stored, which is time consuming. In addition, this process requires us to

balance the resulting *k*d-tree several times during its construction in order to optimize the density estimation computation time. With this method it is difficult to bound the time needed to build the photon map. As for the second method [HHS05], a high number of photons are shot and organized spatially in the *k*d-tree. A photon is picked randomly and density is computed around it using *n*-nearest neighbor search. If the density is higher than the target density, its energy is distributed to the neighboring photons. The photon is invalidated for the next searches in the *k*d-tree. The advantage of this method over the first one is that it is easy to implement with a implicit *k*d-tree, which is built in one go. In addition, the computing time is lower and easily bounded. However this method makes use of a lot of memory and we cannot ensure that the target density is reached for each hit point. Our approach makes use of the second method described above.

A density control function is expressed in photons per unit surface. Let us recall that the zone of influence of a record k is bounded by the intersection between the surface supporting the record and a sphere of radius $a.R_k$ centered at the record position. The minimum area of this intersection is then $\pi.(a.R_k)^2$ (if the surface supporting k is planar). Then, the maximum value of the targeted density D_{kmax} around k is:

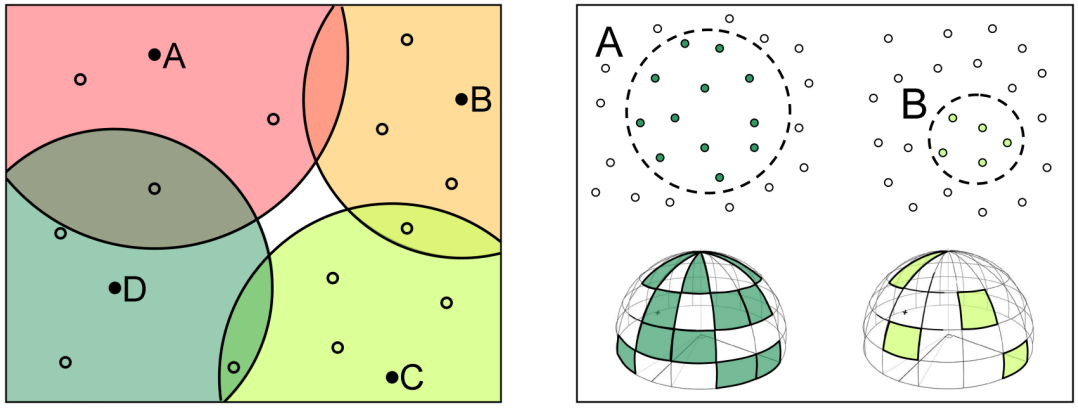
$$D_{kmax} = \frac{n}{a.R_k} \tag{4.1}$$

where n is the maximum desired number of photons within the zone of influence of each record. This density control function provides a photon map meeting the requirements of an accurate and efficient refinement.

We implemented our density control function to generate a photon map following the second method described above. First we create a photon map using the standard approach (with a high number of photons), as well as the associated *k*d-tree. Then, each photon is selected randomly and its associated density is computed using the *n*-nearest neighbors. This computed density is called *actual density* from now on. Then R_k is computed for this photon (Section 4.3.2) using the same *n*-nearest neighbors, which allows to evaluate the corresponding targeted density D_{kmax}. If the actual density is greater than D_{kmax}, then the considered photon is invalidated for the next searches in the *k*d-tree and its power is distributed over the previously found nearest neighbors. Note that only one *n*-nearest neighbors search is needed for each processed photon. Once all the photons have been processed, a new *k*d-tree is built, excluding the invalidated photons. Consequently, the new *k*d-tree is smaller and allows faster searches. It represents the new density controlled photon map.

4.4.2 Discussion

When using density control, the computation times needed in the second pass as well as that for precomputing direct irradiance at photon positions (see data structure in Section 4.3.1) are reduced. However the overall computation time may not be reduced if the time needed by the density control process is higher than the time saved in the rest of the algorithm. We performed some tests, and came to the conclusion that in our case, density control is efficient only in the context of scenes with very complex geometry and/or illumination. For this kind of scene, a high number of photons have to be cast and the resulting photon map may require an



(a) (b)

Figure 4.9: *(a)* When the density of photons on a surface is low, the zones of influence of the records may not to overlap. This results in blank spaces, where irradiance interpolation from the cache is not possible. New records will have to be computed at rendering step to fill these spaces. *(b)* The zone of influence of the record *A* is larger than that of the record *B*. During the refinement pass, more virtual rays are used to fill the hemisphere above *A*, then reducing the number of rays which will be cast.

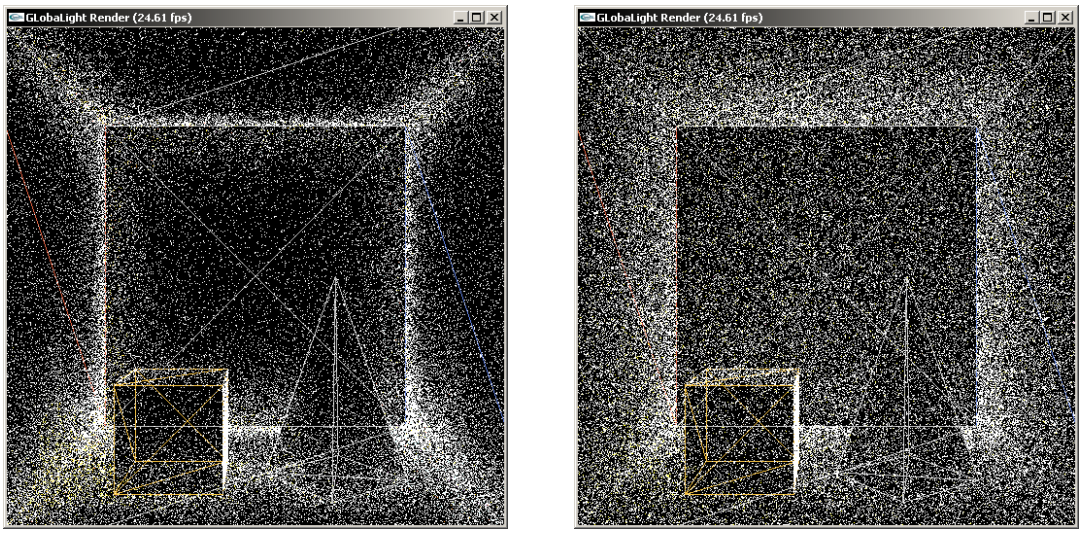
out-of-core memory management. Applying our photon driven irradiance cache algorithm over this type of photon map would require prohibitive computation time. To overcome this problem we propose to split the photon map into several parts which fit into the main memory. We then apply our density control function on each part, and recombine them to obtain a full reduced photon map which can now be stored in-core. Our algorithm can then efficiently generate a high quality irradiance cache.

We have implemented our density control function (using [HHS05]) and the obtained results confirmed the fact that it is efficient only for complex scenes and/or lighting, this is why it is not further discussed in our results.

4.5 Future works

Our current Photon-Driven Irradiance Cache algorithm focuses on computing the indirect diffuse component of the global illumination solution. Although it takes into account the L(D|G)*DE paths (through the use of the photon mapping), we do not store the incoming lighting on glossy surfaces. We could extend our algorithm by generating a *radiance* cache on glossy surfaces too. A coarse approximation of this radiance cache would be quickly derived from the photon map, whereas a more accurate version of the cache, with gradients, could be computed in a following pass. However, computing glossy reflections only from a photon map, without resorting to a prohibitive number of photons, is still a subject of research.

In addition, our current implementation only considers static scenes. Some research work have already been made to adapt the radiance cache to dynamic scenes [GBP06]. Likewise, for photon mapping, interactive construction and visualization of the photon map have been



(a) with density control (b) without density control

Figure 4.10: *(a)* Example of density control for the photon map. A 250k photons photon map has been reduced to 159k photons. *(b)* A photon map containing 159k photons, without density control.

proposed [FD09]. These methods could be combined to yield an algorithm which builds a temporal radiance cache over the scene, from an interactive photon map. The resulting method could be used for interactive walkthrough in dynamic scenes.

4.6 Results

All results were gathered on a Pentium4 3.8Ghz computer with 2GB RAM and an nVidia GeForce 7800GTX 512MB.

4.6.1 General observations

As we use the photon locations as potential positions of irradiance records (Figure 4.12), we cast a high number of photons and account for many light bounces (up to 9 in our test scenes). However our method reuses the photon paths to refine irradiance records during the third pass, hence compensating for the overhead related to the high number of photons and light bounces. The results are given in Tables 4.1 and 4.2.

At the end of the second pass, our algorithm provides an irradiance cache covering most part of the scene. Though unrefined, this coarse cache can be previewed in real time using *radiance cache splatting*. In addition, for each created record, its indirect lighting contribution is distributed among the photons lying within its zone of influence. Note that the resulting photon map (augmented with irradiance values) is similar to that computed in [Chr99], but the irradiance values are computed differently. If we render the scene by tracing only primary

Scene (polygons)	First Pass: **Photon Map Generation**		Second Pass: **Irradiance Cache Generation**	
	time (s)	Nb. photons	time (s)	records
Cornell (32)	0.81	250k (4 bounces)	1.20	2,972
Sponza (66k)	28.3	4M (9 bounces)	11.8	55,649
Sibenik (73k)	29.8	4M (9 bounces)	11.9	73,860

Scene (polygons)	Third Pass: **Refinement** (s)	Total computation time (s)	FPS
Cornell (32)	4.64	6.75	40
Sponza (66k)	123	163	5.6
Sibenik (73k)	162	204	4.7

Table 4.1: Timing of each pass of our method for different scenes. The fps value is the frame rate achieved using *radiance cache splatting* with the computed cache, before and after the refining pass.

Scene	Nb. photons	Number of records
Cornell(32)	250k (4 bounces)	2,972
Sponza(66k)	4M (9 bounces)	55,649
Sibenik(73k)	4M (9 bounces)	77,360

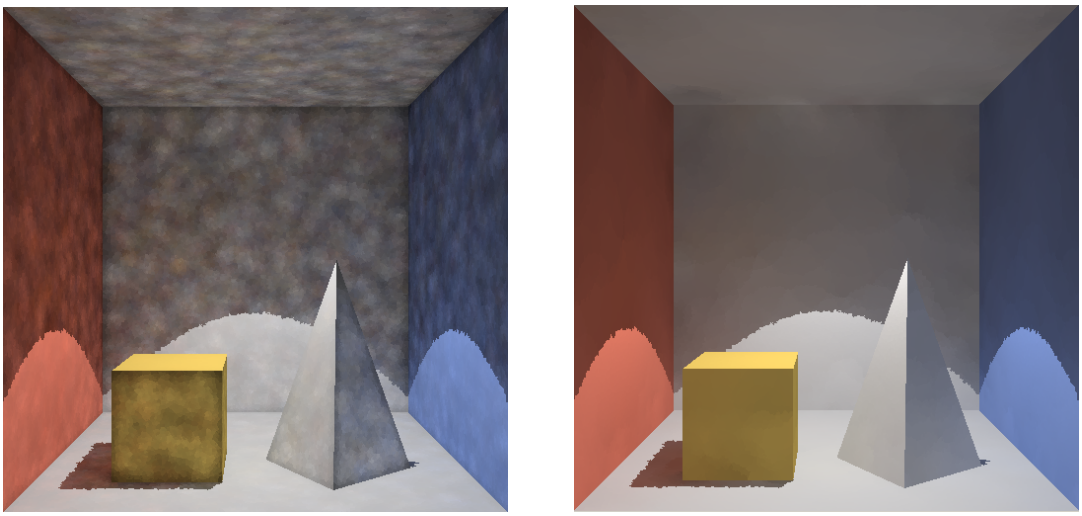
Scene	Classic Final Gathering		Our method		Saved Rays	Saved Time
	time	rays	time	rays		
Cornell(32)	7.14	1,325,335	4.64	802,640	39.4 %	35 %
Sponza(66k)	232	24,205,162	123	11,628,159	52 %	47 %
Sibenik(73k)	285	31,583,736	162	17,006,361	46.2 %	43.2 %

Table 4.2: Performance breakdown. All times are in seconds.

Method	# Records	# Octree queries	# Density estimation	# Nearest photon queries
CNIC	15,783	262,144	2,002,300	6,043,835
VDPDIC (w/o reproj.)	15,905	0	207,517	6,071,735
VDPDIC (with reproj.)	15,905	0	181,360	2,598,815

Method	# Cast rays	Total time for creating the cache
CNIC	8,323,857	145
VDPDIC (w/o reproj.)	8,036,377	96,9
VDPDIC (with reproj.)	4,320,403	57.8

Table 4.3: Results for single view. All times are in seconds.



(a) (b)

Figure 4.11: Images rendered by tracing only primary rays and assigning the irradiance value of the nearest photon to each intersection point. *(a)* Using [Chr99], 150k photons, irradiance values are computed in 1.87 s. *(b)* Our method (end of the second pass), using 150k photons, the computation time of the second pass is 0.89 s.

rays and assigning the irradiance of the nearest photon to each intersection point, our method outperforms [Chr99] in terms of preview quality (Figure 4.11).

4.6.2 Test scenes

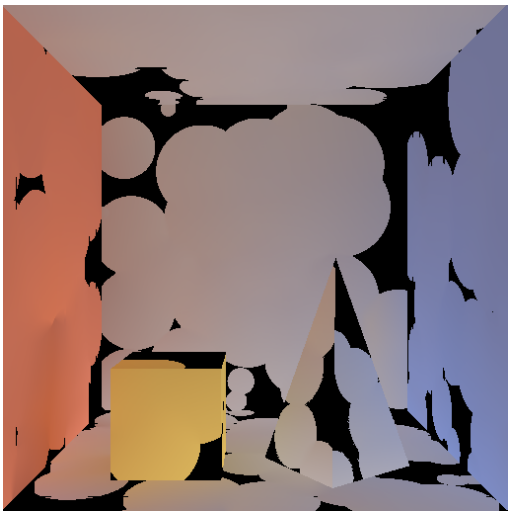
4.6.2.1 Cornell Box

This very simple scene contains one point light source and no textures to illustrate the main ideas behind our method. As shown in Figures 4.12 and 4.13, a high number of photons increases both the coverage of the cache and the quality of the unrefined records while speeding up the refinement process. Table 4.1 shows a breakdown of the rendering times for each step of the algorithm. The resulting cache is generated in 6.75 seconds and contains high quality irradiance values, which can be rendered at 40 fps.

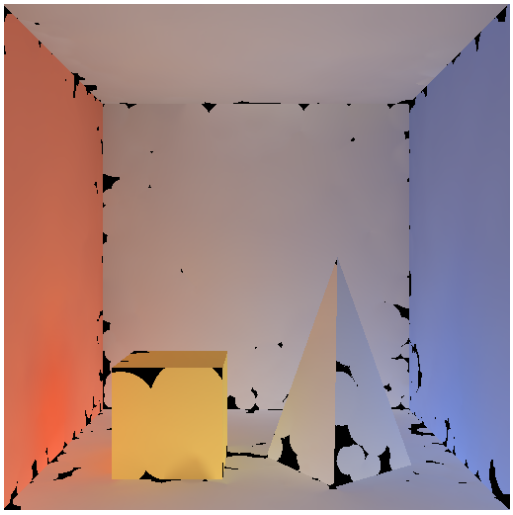
We compared our refinement method used in our third pass with the classical irradiance computation using stratified sampling for the final gathering. As shown in Table 4.2, the reuse of the photon paths allows us to save 39.4% of rays without compromising the rendering quality. We also compared our method with classical per-pixel path tracing. Our method shows no significant error increase compared to the classical irradiance caching algorithm.

4.6.2.2 Sponza Atrium

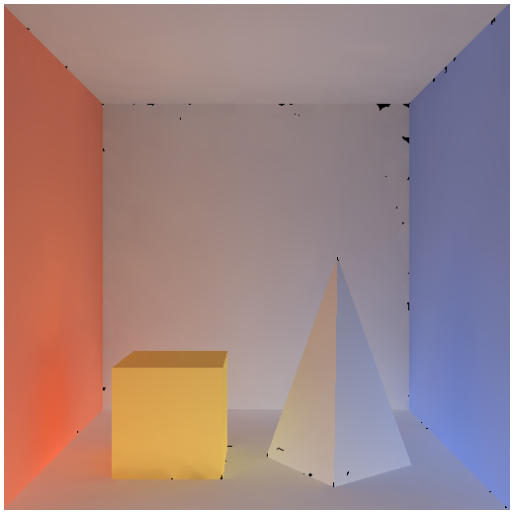
This more complex scene features two point light sources and textures. Our method saves 52% of rays during the third pass, hence speeding up the third pass by 47%. The obtained



(a)

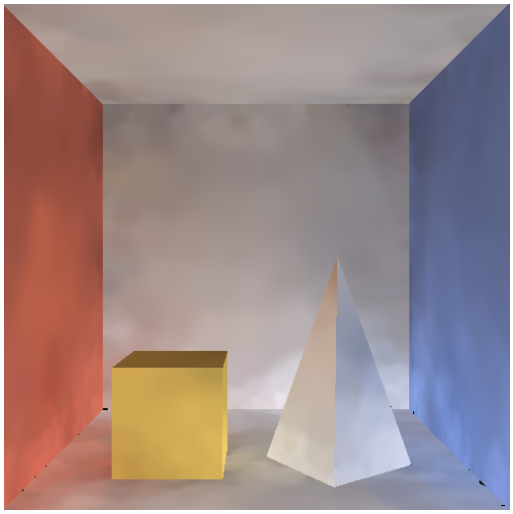


(b)

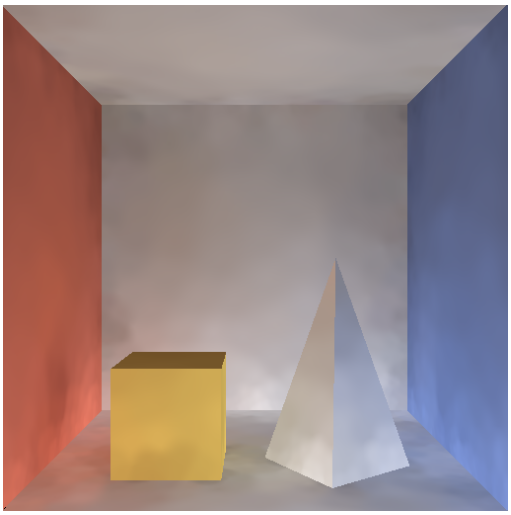


(c)

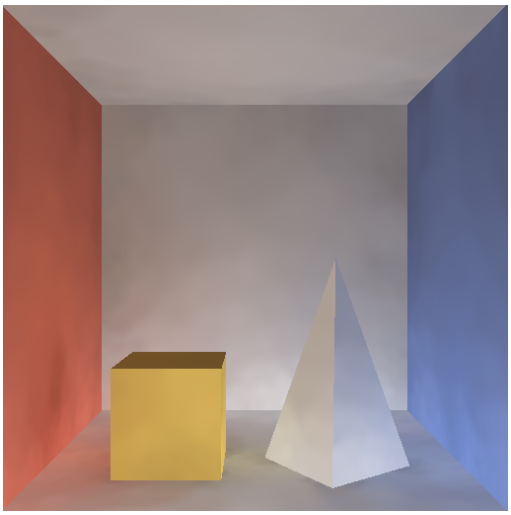
Figure 4.12: Influence of the number of photons on the coverage of the scene by the computed irradiance cache. *(a)* 500 photons. *(b)* 5k photons. *(c)* 50k photons.



(a)



(b)



(c)

Figure 4.13: Influence of the number of photons on the computed coarse irradiance cache. *(a)* 50k photons. *(b)* 150k photons. *(c)* 250k photons: the computed irradiance cache can be used as a good preview of the illumination of the scene.

overall speedup is then 40.1% compared to the use of stratified sampling (final gathering) during the refinement process. The time for computing the irradiance cache is 163 seconds while the rendering frame rate is 5.6 fps.

4.6.2.3 Sibenik Cathedral

This scene contains many small features such as railings and stairs. Therefore, the constructed cache contains more records than the previous scene. The construction of the photon map takes 29.8 s.

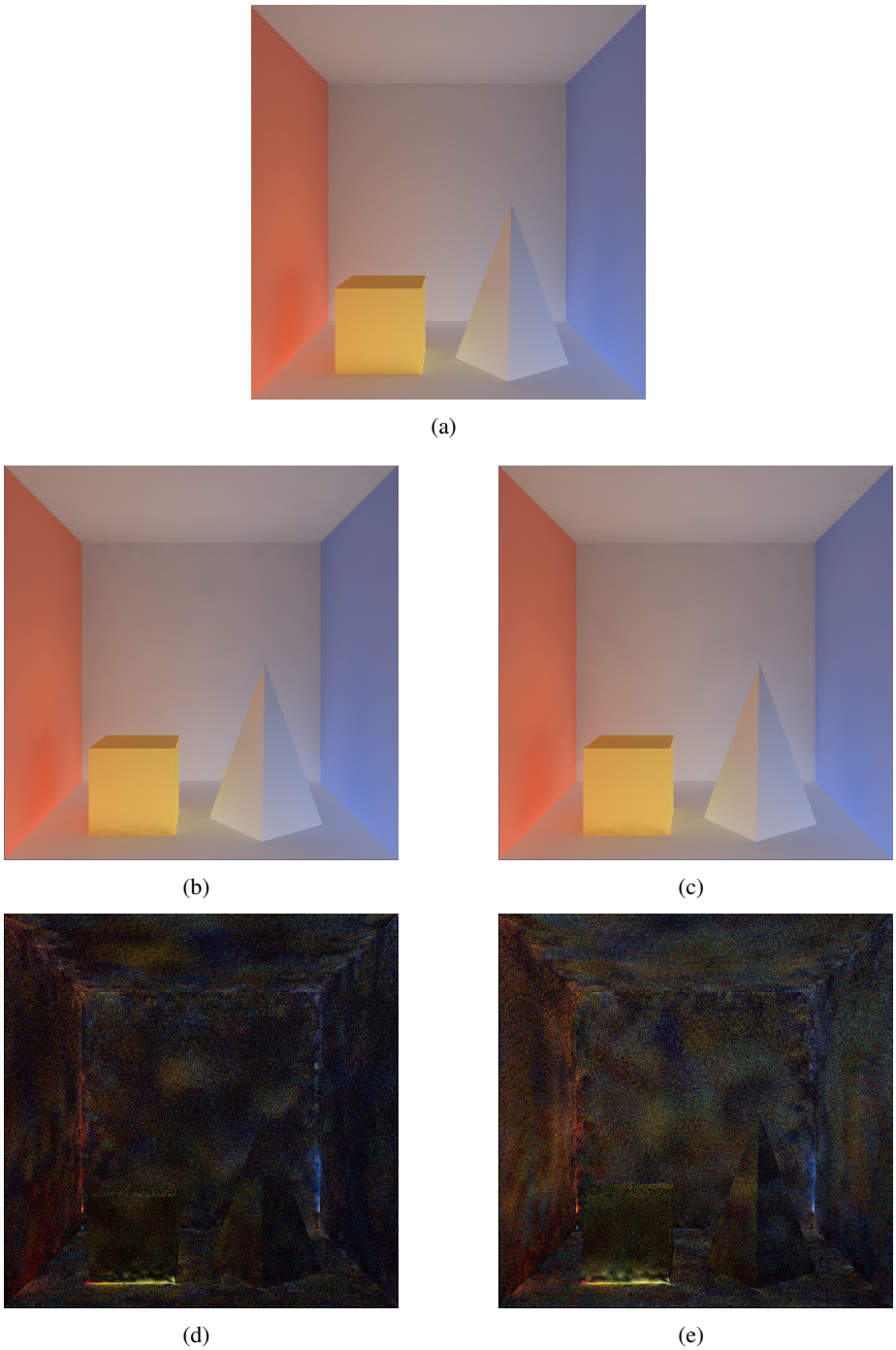
In term of time and quality of preview, we compared two methods: (1) a classical photon mapping algorithm with only primary rays, the irradiance assigned to intersection points being computed using density estimation, (2) our approach (without the third pass) computing records from only the photons within the view frustum. The first approach takes 6.23 seconds while the second requires 2.87 seconds to compute the irradiance cache that can be rendered in real-time with better quality.

As irradiance caching methods are view dependent, we derived a view depend version of our approach to compare it to these methods. Let us call it VDPDIC (View Dependent Photon Driven Irradiance Cache). VDPDIC computes an irradiance cache by using only the visible photons. Let us now consider an irradiance cache method that we call CNIC (Close Neighbor Iradiance Cache). CNIC traces a ray through each pixel and computes (requiring an octree query), when needed, a new record whose irradiance is computed by performing final gathering at the intersection point. Each ray shot by the final gathering process intersects the scene at a point p for which we compute an irradiance value by picking up the one stored at the neighbor photon. For the two methods the image resolution is 512*512, and the number of shot rays for each final gathering is 384. The results obtained for these two methods are given by Table 4.3. Using reprojection in VDPDIC allows to get a saving of 40% for irradiance cache construction as well as for final gathering rays. Finally, VDPDIC is 2.5 faster than CNIC. Recall that this comparison concerns only a single view, while our method is capable of computing a single cache for all views.

4.7 Conclusion

We proposed a method for building an irradiance cache from a photon map. The two main advantages of our approach is that it makes possible good quality and fast preview of an existing photon map and, unlike [WRC88], it generates an irradiance cache that is view-independent, say it covers the entire scene. Our algorithm consists of three passes: (1) construction of the photon map, (2) creation of records, (3) refinement of the cached irradiance values. Radiance cache splatting [GKBP05] can be used for fast preview of the global illumination solution (end of the second pass) and for high quality interactive rendering (end of the third pass). We have also proposed and implemented a density control function for photon map construction. After many experiments we ascertained that it is efficient only for complex scenes needing high number of photons as it may avoid out-of-core memory management by drastically reducing the size of the generated photon map. To make our approach more efficient, one solution is to either improve the density control process (while keeping the same density function) or

propose a more efficient method of guiding the photons emitted from the light sources. Finally, our future works includes the extension to glossy surfaces and dynamic scenes.



(a)

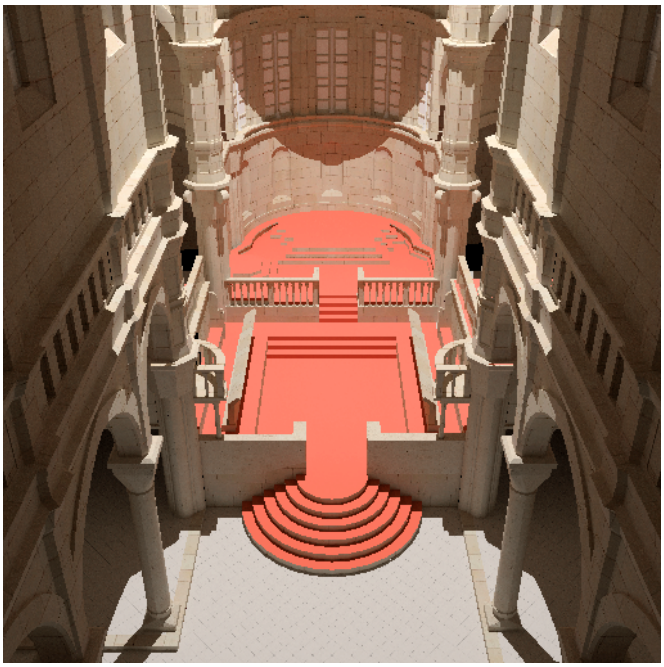
(b)

(c)

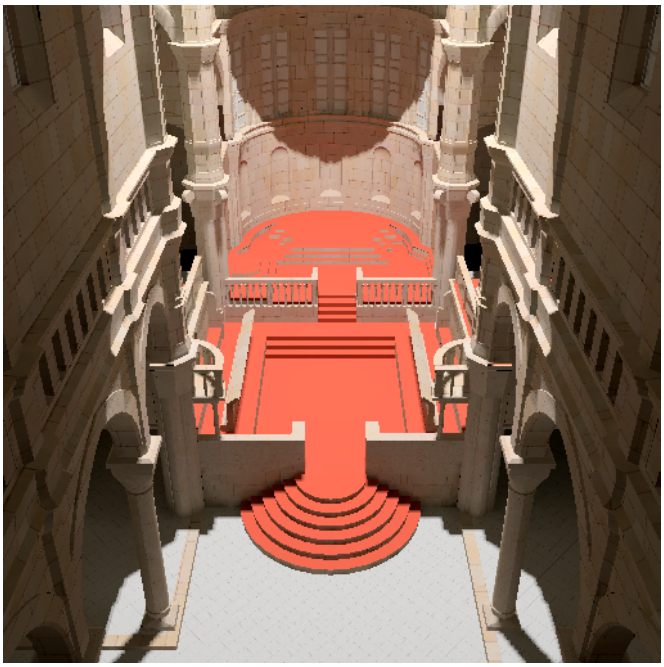
(d)

(e)

Figure 4.14: *(a)* Raytraced reference solution, 900 paths per pixel, 2500 s. *(b)* Standard irradiance caching algorithm. *(c)* Our method, with reuse of rays during refinement. *(d)* Differences between (a) and (b). *(e)* Differences between (b) and (c). These differences are multiplied by 15.



(a)



(b)

Figure 4.15: Sibenik Cathedral. *(a)* Pure Monte Carlo, 3 hours. *(b)* Our method, 163 seconds, including photon map and cache construction. Rendering is interactive.

Chapter 5

A Bayesian Monte Carlo approach to Global Illumination

5.1 Introduction

Monte Carlo integration has now become an essential tool in the field of global illumination. It allows accurate simulation of light transport mechanisms and leads to photorealistic images. However, its computational cost remains high due to the huge number of ray-traced samples that need to be collected before reaching an acceptable noise level. This problem has given rise to an extensive literature of which many of the proposed solutions are based on importance sampling and/or (to a less extent) control variates. Both techniques try to exploit some prior knowledge of the integrand so as to build an approximating function. This approximating function is then used either to optimize samples distribution (importance sampling) or to extract a known deterministic part of the integrand so as to reduce its variance.

In [O'H87], O'Hagan raises two fundamental objections to the importance sampling procedure. First, the estimator depends on the arbitrary choice of the sampling density. Therefore, the same set of observed integrand sample values will lead to different estimates depending on the chosen proposal distribution, which violates the Likehood Principle [Bir62]. Briefly, the Likehood Principle states that "in the inference about θ, after x is observed, all relevant experimental information is contained in the likelihood function ($p(x|\theta)$) for the observed x". Consequently, the rules governing the data collection process are irrelevant for computing the estimate. Even though the choice of the sampling density is guided by prior information we have on the integrand, the objection still holds as it remains some freedom on the choice of the sampling density.

The second objection is that Monte Carlo procedures ignore the sample locations, only the sample values of the integrand are used. Therefore, two samples falling on the same or close locations will have equal importance in the estimation process, whereas the second sample brings no extra information. Of course, such occurrences can be made unlikely with stratification or even impossible with quasi-Monte Carlo (QMC) but nonetheless, the fact remains that classical Monte Carlo wastes important information.

To avoid these inconsistencies and allow better estimates, O'Hagan turns the problem of

evaluating the integral into a Bayesian inference problem [O'H91]. To this end, he assumes a Gaussian Process (GP) behavior for the integrand which leads to a new form of quadrature called "Bayes-Hermite quadrature". His results have been further generalized and applied in various domains related to Machine Learning theory under the generic name of "Bayesian Monte Carlo" (BMC) [PHR06, KNKS08].

The goal of this chapter is to show how the Bayesian Monte Carlo method can be applied to global illumination models while trying to obtain the best compromise between computational complexity and effective noise reduction. However, as BMC significantly departs from conventional MC, it would be too long to analyze and compare in details all theoretical aspects and implementation options in this chapter. Besides, many research directions need to be investigated before assessing the full potential of this new approach. Therefore, our presentation of the theoretical background only covers the minimum necessary to understand the application to global illumination and, although only one implementation method is described, we will briefly present alternative approaches.

We will first present a brief introduction to the theoretical background of Bayesian Monte Carlo integration and discuss in general the important implementation issues and options. Then, we will present a first attempt to apply BMC to compute the integral involved in the final gathering phase of a photon mapping algorithm. We will show how a GP model can be built in this particular case and how the quadrature coefficients can be derived from this model. We will also address the problem of optimal sampling and hyperparameters estimation, and present our solutions to these problems. As we will see, we have obtained very promising results despite suboptimal hyperparameters.

5.2 Related work

Although the theory of Bayesian Quadrature has been known since 1991 [O'H91], it is only recently that it has been put into practice (e.g. [PHR06, KNKS08]) and to the authors' knowledge, no research in this direction has yet been reported in the computer graphics literature. In the field of global illumination, most research works are based on importance sampling and although this variance reduction technique can be applied to many cases (e.g. [CAE08]), its efficiency is questionable when prior information does not allow to find a well-focused proposal distribution. This is particularly noticeable when diffuse reflection is involved since the cosine distribution law gives rise to rather scattered sampling directions.

The control variates method has been proposed to supplement importance sampling by introducing a correlated function which approximates the integrand with a constant difference term [LW94]. In multiple importance sampling (MIS), the control variate estimate is built with the same component functions as the ones used in the mixture probability density functions [OZ00, FCH+06, Vea98]. When control variates is used without importance sampling, finding the optimal mixture coefficients amounts to a least square fitting of the integrand as shown in [HLO04]. There are some similarities with Bayesian Monte Carlo with regard to this point since BMC also involves a regression step. However, as discussed in [RW06], Bayesian regression differs in that it is a non-parametric method, which means (in brief) that the basis functions of the feature space and their associated weights do not appear explicitly. Consequently, the

computational complexity does not depend on the dimension of the feature space but on the size of the sample set only. Actually, the dimension of the feature space implicitly involved in a Bayesian regression may be infinite.

5.3 Issues at stake

In [RG02], Rasmussen and Ghahramani have shown that Bayesian Monte Carlo can significantly outperform any classical importance sampling method if appropriate prior knowledge is used. However, such ideal conditions are rarely met in the context of global illumination. The main reason is that the Gaussian Process (GP) prior used in the BMC method is very effective in reducing the estimator variance when the integrand is a smooth function of the input variables. This smooth integrand behavior is rare in global illumination models mainly because of the sharp changes of illumination at object boundaries [DLAW01]. We will see that the features of illumination statistics can be taken into account in the GP model so as to alleviate the effects of discontinuities. Surprisingly, we have already obtained good results with a very simple GP model and largely suboptimal hyperparameters, which raises hopes of wide margin for improvement.

5.4 Background

5.4.1 Notation

A generic point in the D dimensional unit cube $[0,1)^D$ is denoted by $\mathbf{x} = (x^1, \ldots, x^D)^t$ while a random point in the same space is denoted $\mathbf{X} = (X^1, \ldots, X^D)^t$ and a point used in an integration rule is $\mathbf{X}_i$. In conventional Monte Carlo methods, $\mathbf{X}_i$ is a sample of a random vector $\mathbf{X}$ drawn from a given density.

5.4.2 Bayesian quadrature equations

We consider the problem of computing the integral I of the product of a function $f(\mathbf{x})$ with a weight function $p(\mathbf{x})$, both functions defined on the D dimensional unit cube $[0,1)^D$:

$$I = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \qquad \text{with } \mathbf{x} \in [0,1)^D \tag{5.1}$$

Here and elsewhere, integrals without explicit range are understood to be over $[0,1)^D$. Typically, $p(\mathbf{x})$ is known under its analytical form whereas $f(\mathbf{x})$ can only be evaluated numerically or through computer simulation. $p(\mathbf{x})$ is not required to be a probability density function, however we will assume that it is normalized:

$$\int p(\mathbf{x})d\mathbf{x} = 1$$

In Bayesian Monte Carlo (BMC) the problem of evaluating the integral in Equation (5.1) is turned into a Bayesian inference problem, which allows avoiding the inconsistencies discussed

in section 5.1 and leads to a better estimate. As proposed by O'Hagan in [O'H91], BMC is based on the following reasoning: $f(\mathbf{x})$ is considered as random simply because it is unknown (and thus uncertain) before its evaluation. This view may seem to be somewhat inappropriate but it is totally consistent with the Bayesian approach that all forms of uncertainty can be modeled by probabilities. For this purpose, we will put a prior on $f(\mathbf{x})$ by using a Gaussian Process (GP). Detailed presentation of GPs is beyond the scope of this chapter, a comprehensive introduction to GPs can be found in [RW06]. Formally, a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is completely defined by its mean function $\bar{f}(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$ which depends only on the input $\mathbf{x}$, ie. the sample locations:

$$\begin{aligned} \bar{f}(\mathbf{x}) &= \mathrm{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathrm{E}[(f(\mathbf{x}) - \bar{f}(\mathbf{x}))(f(\mathbf{x}') - \bar{f}(\mathbf{x}'))] \end{aligned} \tag{5.2}$$

and will be denoted by:

$$f(\mathbf{x}) \sim \mathcal{GP}[\bar{f}(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')]$$

The choice of the covariance function $k(\mathbf{x}, \mathbf{x}')$ allows us to introduce prior knowledge on the smoothness properties of the integrand. However, this choice is not arbitrary as the covariance function must be a symmetric ($k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$) and positive semidefinite kernel (see [RW06] p. 80 for details). A covariance function is stationary when it is invariant to translation, ie. $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$. Moreover, the covariance function is isotropic when it is a function only of $r = |\mathbf{x} - \mathbf{x}'|$. In this case, these are also known as *radial basis functions* (RBFs). Furthermore, Bochner's theorem states that a stationary covariance function is positive semidefinite if and only if its Fourier transform is non-negative [RW06].

A common choice of the stationary covariance function is the squared exponential (SE) covariance function:

$$k(\mathbf{x}_1, \mathbf{x}_2) = w_0^2 \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_1^{(d)} - x_2^{(d)}}{w_d}\right)^2\right\} \tag{5.3}$$

where the w_i's are the hyperparameters of the model. With this function, close function samples are highly correlated whereas $k(\mathbf{x}_1, \mathbf{x}_2) \approx 0$ for distant samples, which means that the function values $f(\mathbf{x}_1)$ and $f(\mathbf{x}_2)$ are almost independent. Let us note that $w_0^2 = k(\mathbf{x}, \mathbf{x}) = \mathrm{Var}[f(\mathbf{x})]$. The other hyperparameters $w_1, \ldots, w_D$ are the lengthscales of the individual input dimensions. The SE covariance function is thus isotropic when all the lengthscales are equal. A GP having a SE covariance function has mean square derivatives of all orders and is thus very smooth. This strong smoothness might not be appropriate for illumination models. The Matérn class [RW06] of covariance functions have been recommended by Stein when such smoothness is unrealistic. Although we have already obtained good results with SE covariance function in the application described below, we will consider the Matérn class option in future works.

Now that we have specified the GP prior that models our beliefs about $f(\mathbf{x})$, let us suppose that we are provided with a set of noisy samples $\mathcal{D} = \{(\mathbf{X}_i, Y_i) \,|\, i = 1, \ldots, n\}$ where Y_i is the observed value of f at point $\mathbf{X}_i$:

$$Y_i = f(\mathbf{X}_i) + \varepsilon_i$$

and the ε_i are samples of an independent, identically distributed Gaussian distribution with zero mean and variance σ^2. Then, in application of the Bayes' rule, the posterior process is also a GP with mean and covariance given by [RW06]:

$$\begin{aligned} \mathrm{E}[f(\mathbf{x})|\mathcal{D}) &= \bar{f}(\mathbf{x}) + \mathbf{k}(\mathbf{x})^t Q^{-1}(\mathbf{Y} - \bar{\mathbf{F}}) \\ \mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')|\mathcal{D}] &= k(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^t Q^{-1}\mathbf{k}(\mathbf{x}') \end{aligned} \tag{5.4}$$

where:

$$\begin{aligned} \mathbf{k}(\mathbf{x}) &= (k(\mathbf{X}_1, \mathbf{x}), \ldots, k(\mathbf{X}_n, \mathbf{x}))^t \\ Q &= (K + \sigma^2 I_n) \\ K_{i,j} &= k(\mathbf{X}_i, \mathbf{X}_j) \qquad \text{with } (i, j) \in [1, n]^2 \\ \mathbf{Y} &= (Y_1, \ldots, Y_n)^t \\ \bar{\mathbf{F}} &= \left(\bar{f}(\mathbf{X}_1), \ldots, \bar{f}(\mathbf{X}_n)\right)^t \end{aligned}$$

where I_n is the identity matrix. Equation (5.4) gives the expected value (or the mean prediction) of f for an unseen input $\mathbf{x}$ given the observed data $\mathcal{D}$. This particular form of regression is called *Bayesian regression*. When the covariance function is a RBF, Bayesian Regression is not to be confused with RBF non-linear regression (see [Her04, RW06] for more details).

From this, we can derive the posterior distribution of the integral I given by Equation (5.1). As integration is a linear operation, the distribution of I is Gaussian with mean and variance [O'H91]:

$$\mathrm{E}(I|\mathcal{D}) = \bar{I} + \mathbf{z}^t Q^{-1}(\mathbf{Y} - \bar{\mathbf{F}}) \tag{5.5}$$

$$\mathrm{Var}(I|\mathcal{D}) = \bar{V} - \mathbf{z}^t Q^{-1}\mathbf{z} \tag{5.6}$$

where:

$$\begin{aligned} \bar{I} &= \int \bar{f}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ \mathbf{z} &= \int \mathbf{k}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ \bar{V} &= \iint k(\mathbf{x}, \mathbf{x}') p(\mathbf{x}) p(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \end{aligned} \tag{5.7}$$

The estimate of I, given our GP prior and the observed data $\mathcal{D}$, is then $\hat{I}_{BMC} = \mathrm{E}(I|\mathcal{D})$. Note that $\bar{I}$ and $\bar{V}$ are respectively the prior mean and variance of I, ie. the $\hat{I}_{BMC}$ estimate and its variance when $n = 0$ in Equation (5.5) and (5.6).

The $\mathbf{z}$ vector can also be expressed in terms of the function $f_z(\mathbf{x}')$ given by:

$$f_z(\mathbf{x}') = \int k(\mathbf{x}, \mathbf{x}') p(\mathbf{x}) d\mathbf{x} \tag{5.8}$$

Then we have:

$$\mathbf{z} = (f_z(\mathbf{X}_1), \ldots, f_z(\mathbf{X}_n))^t$$

and Equation (5.7) can be rewritten as:

$$\bar{V} = \int f_z(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{5.9}$$

In Equation (5.6), the estimate $\mathrm{Var}(I|\mathcal{D})$ is based only on prior expectation since it depends on sampling points $\mathbf{X}_i$ but not on the sample values Y_i.

Equation (5.5) can also be rewritten as:

$$\mathrm{E}(I|\mathcal{D}) = \bar{I}_0 + \mathbf{c}^t \mathbf{Y} \tag{5.10}$$

where:

$$\bar{I}_0 = \bar{I} - \mathbf{c}^t \bar{\mathbf{F}} \tag{5.11}$$

$$\mathbf{c} = Q^{-1} \mathbf{z} \tag{5.12}$$

Equation (5.10) expresses a quadrature rule in which $\mathbf{c}$ is the vector of quadrature coefficients. This form of quadrature is called *Bayesian quadrature* and *Bayes-Hermite quadrature* when $p(\mathbf{x})$ is a Gaussian distribution. In this latter case or when $p(\mathbf{x})$ is uniform, the quadrature coefficients can be computed through a closed-form solution [O'H91].

5.4.3 Overview of the Bayesian Monte Carlo method

The first problem that we have to solve in implementing the Bayesian Monte Carlo method is how to choose the mean function and the covariance function of the GP model associated with $f(\mathbf{x})$. The first point is addressed in the following section and we have already discussed in the above section the general properties of covariance functions and their impact on the smoothness assumption. The problem of hyperparameters selection is specifically addressed in section 5.4.3.3. Given a set of samples $\mathcal{D}$, we will then need to compute the vector of quadrature coefficients $\mathbf{c}$ with Equation (5.12) and the I estimate with Equations (5.11) and (5.10). The main computing task is represented by Equation (5.12) which involves the inversion of the Q matrix and the computation of the $\mathbf{z}$ vector using Equation (5.8). We will see in the following how to solve this problem for our application. Let us recall that with the BMC method, samples can be drawn from arbitrary distributions but the sampling strategy has a significant effect on the variance as discussed in section 5.4.3.2 and 5.5.4.

5.4.3.1 Finding a mean function

The GP prior assumes that we know a mean function $\bar{f}(\mathbf{x})$. This mean function does not need to be an accurate estimate of the statistical mean at any point $\mathbf{x}$. A rough approximation of $f(\mathbf{x})$ is often sufficient since the generated bias is negligible with the sample sets sizes used in practice. In the context of global illumination, well-known CG techniques can be used to determine such a function (e.g. radiance cache, spherical harmonics, etc.). Depending on how well $\bar{f}(\mathbf{x})$ can model the discontinuities of $f(\mathbf{x})$, the high frequency components of the difference $f(\mathbf{x}) - \bar{f}(\mathbf{x})$ can be reduced, which allows the smoothing kernel of the GP model to more faithfully fit the observed data statistics. Note also that there are some similarities with

the approximating function used in the control variates method and techniques such as the 5D tree proposed in [LW95] can be used as well.

Another alternative offered by the GP theory is to infer $\bar{f}(\mathbf{x})$ from the observed data $\mathcal{D}$ as proposed in [RW06]. To this end, we write $\bar{f}(\mathbf{x})$ as:

$$\bar{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^t \beta$$

where $\mathbf{h}(\mathbf{x})$ is a given set of m basis functions (such as spherical harmonic basis) and β are unknown parameters. The derivation of the vector of coefficients β can be found in [RW06]. This solution results in an additional variance term that decreases rapidly as n increases. For the application described in this chapter, we chose a constant mean function. We will see that this simple solution gives good results for the sizes of the sample sets we used. However, it is possible that a more refined model of the mean function as well as adaptive techniques would be required to decrease the sizes of the samples sets (cf. discussion in section 5.7.4).

5.4.3.2 Optimal sampling

One important advantage of Bayesian Monte Carlo over importance sampling is that samples do not need to be drawn from predefined distributions. However, some distributions will be more efficient than others. The space filling designs of quasi-Monte Carlo (QMC) methods [Nie92, HLO04] can be used but their optimality criteria correspond to special cases of Bayesian optimal designs and are mostly based on intuitive considerations. A more appropriate technique consists in choosing the sampling points $\mathbf{X}_1, \ldots, \mathbf{X}_n$ to minimize the expected variance $\mathrm{Var}(I|\mathcal{D})$ [O'H91, Min00]. We have used this method in our work (cf. 5.5.4). Note however that this design will be based on prior expectation only.

5.4.3.3 Adaptation of hyperparameters

The GP prior defined above assumes that we know the hyperparameters of the model. In the case of the SE covariance function (Equation (5.3)), the vector of hyperparameters is:

$$\vartheta = (\sigma, w_0, w_1, \ldots, w_D)^t$$

where σ^2 is the variance of the additive noise. The choice of these hyperparameters must reflect our beliefs on the variability and the coherence of the function $f(\mathbf{x})$ to be integrated. In [RW06], Rasmussen suggests a maximum likehood solution that consists in finding ϑ that maximizes the marginal likehood:

$$\hat{\vartheta} = \underset{\vartheta}{\mathrm{argmax}}[p(\mathcal{D}|\vartheta)]$$

The marginal likehood $p(\mathcal{D}|\vartheta)$ is the probability of the observed data $\mathcal{D}$, given the model hyperparameters ϑ. Rasmussen expresses the logarithm of the likehood function with the following equation:

$$\log p(\mathcal{D}|\vartheta) = -\frac{1}{2}\mathbf{Y}^t Q^{-1}\mathbf{Y} - \log|Q| - \frac{n}{2}\log 2\pi$$

However, finding the set of hyperparameters $\hat{\vartheta}$ which maximizes this log marginal likehood is very costly. Indeed, this maximization has to be performed independently for each sample set $\mathcal{D}$ and the associated $\mathbf{Y}$. In addition, the log marginal likehood evaluation requires to compute Q and its inverse. Since the likehood has to be evaluated several times to reach a maximum, finding ϑ for each set of data is very computationally expensive.

Consider a problem where there it is not possible to gather more data. In this case, finding an optimal set of hyperparameters $\hat{\vartheta}$ may be the only way to improve the accuracy of the Bayesian Monte Carlo estimate. Hence the expensive computation cost is generally acceptable. However, in the case of global illumination, it is generally possible to obtain more data, *i.e.* cast more rays. The computation cost needed to determine $\hat{\vartheta}$ have to be balanced with the cost of casting additional rays to achieve the same variance reduction. Most part of the time, it is more profitable casting additional rays rather than finding $\hat{\vartheta}$.

Another solution is to find the hyperparameters that best fit the real covariance function built from a set of training data. This can be done at a global level for the whole scene or a specific view but this choice may be locally suboptimal. We have chosen this last solution in the application presented in this chapter. However a simplified version of the maximum likehood solution will be investigated in future works, as introduced in Section 5.6.1.

5.5 Application to final gathering

5.5.1 The irradiance integral

We have chosen final gathering as a case study to show the benefit of Bayesian Monte Carlo because it is typically a case for which basic Monte Carlo integration methods work poorly. As regards importance sampling, the only adequate proposal distribution that can be extracted from the analytical part of the integrand is the cosine distribution which leads to highly scattered samples. This situation does not allow us to faithfully represent the incident radiance. This problem is solved by the BMC approach which uses a GP to model the radiance variations between sample points, and consequently yields a better estimate of the integral. Of course, in the Monte Carlo method, more adequate distributions can be built from observed data as in [CAE08]. As for BMC, equivalent techniques (such as active learning) exist but they are difficult to implement in this context.

Briefly, the final gathering is the final step of the photon mapping method. It consists in computing the irradiance of visible surfaces by casting rays from these surfaces so as to capture the distribution of incident light. The irradiance at a given point P of a surface S is given by the integral:

$$E_s(P) = \int_{\Omega_{2\pi}} L(\theta, \phi) \cos\theta \, d\Omega \tag{5.13}$$

where θ and ϕ define the direction of a ray of incident light, $d\Omega$ is the elementary solid angle, $L(\theta, \phi)$ is the radiance of the surface intersected by this ray and $\Omega_{2\pi}$ is the solid angle representing the hemisphere centered at P and located above the considered surface. The couple (θ, ϕ) defines the spherical coordinates of a point η on the unit sphere centered at P. The origin of this coordinate system is P and its z axis coincides with the normal N to the surface at point P.

By expanding $d\Omega$ in terms of (θ, ϕ), we obtain:

$$E_s(P) = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} L(\theta, \phi) \cos\theta \ \sin\theta \, d\theta d\phi \tag{5.14}$$

To obtain an integral in the same form as Equation (5.1), we rewrite Equation (5.14) as:

$$E_s(P) = \pi \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} L(\theta, \phi) p(\theta, \phi) \, d\theta d\phi \tag{5.15}$$

where:

$$p(\theta, \phi) = \frac{\cos\theta \sin\theta}{\pi} = \frac{\sin 2\theta}{2\pi} \tag{5.16}$$

is a normalized weight function.

5.5.2 A Gaussian Process model for incident illumination

As discussed in section 5.4.3, the first step to implement the Bayesian Monte Carlo method is to define the GP model associated with the "unknown" part of the integrand of Equation (5.15), namely the GP mean and covariance functions associated with the incident radiance $L(\theta, \phi)$. The role and the properties of these functions has already been presented in section 5.4.2 and 5.4.3.1. Our choice of mean function will be discussed in section 5.5.7.1. As regards the covariance function, it must be isotropic (all dimensions are equivalent) and positive definite on the unit sphere. Several functions having these properties have been proposed in the literature (e.g. [RW06]). We have obtained good results with the isotropic form of the SE covariance function defined in Equation (5.3) despite its strong smoothing effect. It is given by the following equation:

$$k(r) = w_0^2 \exp\left(-\frac{r^2}{2l^2}\right) \tag{5.17}$$

where r is the Euclidean distance between two points η_1 and η_2 on the unit sphere, and l is the lengthscale hyperparameter. Let us call d the geodesic distance (i.e. the angle between the directions defined by η_1 and η_2). Then we have:

$$d = \arccos(\eta_1 \cdot \eta_2) \tag{5.18}$$

and:

$$\begin{aligned} r &= \sqrt{2 - 2\eta_1 \cdot \eta_2} = 2 \sin\frac{d}{2} \qquad 0 \leq d \leq \pi \\ r^2 &= 2(1 - \cos d) \end{aligned} \tag{5.19}$$

Using Equation (5.19), the covariance function given by Equation (5.17) can be expressed as a function of the geodesic distance d:

$$k(d) = w_0^2 \exp\left(\frac{\cos d - 1}{l^2}\right) \tag{5.20}$$

Equation (5.18) can be expressed in terms of spherical coordinates as follows:

$$\cos d = \sin\theta_1 \sin\theta_2 \cos(\phi_2 - \phi_1) + \cos\theta_1 \cos\theta_2 \tag{5.21}$$

where (θ_1, ϕ_1) and (θ_2, ϕ_2) are the spherical coordinates of η_1 and η_2 respectively. The covariance function can then be expressed as a function of the spherical coordinates of η_1 and η_2 as follows:

$$k(\eta_1, \eta_2) = w_0^2 \exp\left(\frac{\sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2) + \cos\theta_1 \cos\theta_2 - 1}{l^2}\right) \tag{5.22}$$

5.5.3 Determination of the quadrature coefficients

Equation (5.22) allows us to build the covariance matrix Q for a set of sampling directions $\mathcal{D} = \{\eta_1, \ldots, \eta_n\}$.

To compute the quadrature coefficients given by Equation (5.10), we also need to compute the coefficients $z_i = f_z(\eta_i)$ of the $\mathbf{z}$ vector for each sample η_i of $\mathcal{D}$. Using Equation (5.16), (5.20) and (5.21), the expression of the f_z function of Equation (5.8), becomes:

$$f_z(\eta_i) = f_z(\theta_i, \phi_i) = \frac{1}{2\pi} \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} k(\eta, \eta_i) \sin 2\theta \, d\theta d\phi \tag{5.23}$$

where (θ_i, ϕ_i) are the spherical coordinates of a sampling direction η_i. Let us observe that in Equation (5.21), the variable ϕ_i only appears in the $\cos(\phi - \phi_i)$ term. Since the integration domain covers a whole period of the $\cos(\phi - \phi_i)$ function, z_i is independent of ϕ_i:

$$f_z(\theta_i, \phi_i) = f_z(\theta_i, 0) = f_z(\theta_i)$$

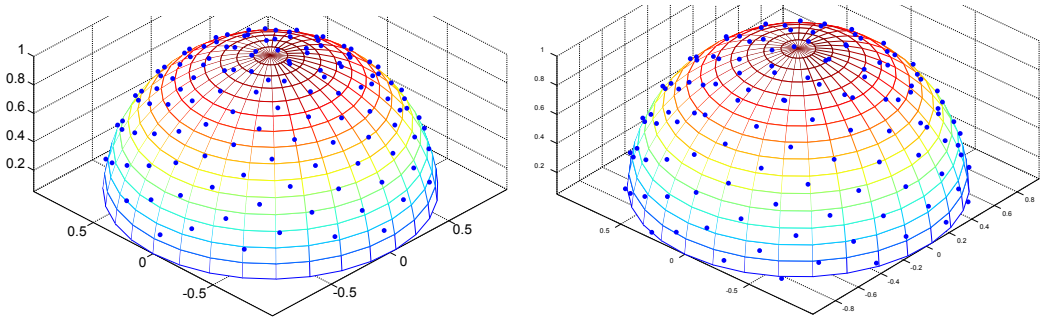
Moreover, since the dependence of the covariance function as regards ϕ_1 and ϕ_2 is only contained in the term $\cos(\phi_2 - \phi_1)$ (cf. Equation (5.20) and (5.21)), the covariance matrix Q is unchanged if the set of sampling directions $\mathcal{D} = \{\eta_1, \ldots, \eta_n\}$ is rotated around the z axis. As the z_i coefficients depend only on θ_i, this means that the whole vector of quadrature coefficients $\mathbf{c} = Q^{-1}\mathbf{z}$ is also unchanged. This property will be very useful to generate different sample sets while keeping the same vector of quadrature coefficients.

5.5.4 Optimal sampling for the diffuse component

The variance function can be computed from (5.6). In this equation, $\bar{V}$ can be derived from Equation (5.9) and (5.23):

$$\bar{V} = \frac{1}{2\pi} \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} f_z(\theta) \sin 2\theta \, d\theta d\phi = \int_{\theta=0}^{\frac{\pi}{2}} f_z(\theta) \sin 2\theta \, d\theta$$

As mentioned in section 5.4.3.2, optimum sample sets for given values of l, σ, w_0 and n (the size of the sample set) can be computed by finding the samples locations that minimize the variance function of Equation (5.6). Compared to a uniform distribution generated by the spiral points



(a) lengthscale = 0.1 (b) lengthscale = 0.2

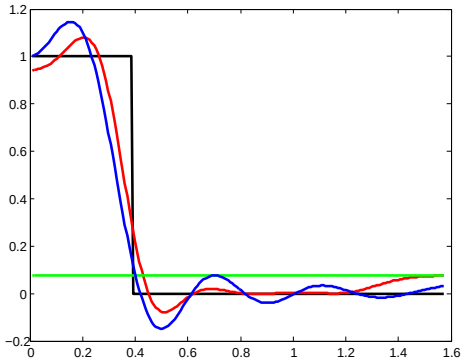
Figure 5.1: Optimized sample set of 128 points

algorithm of Rakhmanov et al [SK97], optimized sets roughly bring a 6 dB variance reduction (as computed from Equation (5.6)).

As expected, the minimization process becomes very computationally demanding when n reaches one hundred or more samples. To speed up the computation, we only specify the θ values as input arguments to the variance function. This choice is sensible since the distribution in θ is dominant given the respective roles of θ and ϕ in the irradiance integral. If we pursue this line of reasoning further, we can choose to act on a global θ distribution function rather than on individual θ coordinates of samples. If this distribution function is modeled by a polynomial of degree p, this will reduce the number of input variables to $p + 1$. To generate the samples locations, we use a slightly modified version of the spiral points algorithm:

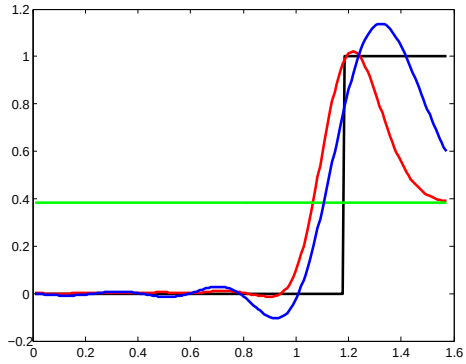
```
dphi = pi*(3-sqrt(5));
phi = 0.;
dz = 1./n;
z = 1- dz/2;
for {k = 1:n}{
    zv = polyval(dcoeff,z); % z coord.
    thetas(k) = acos(min(zv,1)); % Theta
    phis(k) = mod(phi,2*pi); % Phi
    z = z -dz;
    phi = phi + dphi;
end
```

In the standard spiral points algorithm, the z coordinate of points are uniformly distributed. In our algorithm, this linear distribution is modified by the polynomial function *polyval*(). The input variables for the optimizer are then the coefficients *dcoeff* of the polynomial. The algorithm is initialized with the uniform distribution, i.e. the monomial $y = x$. A degree 4 or 5 is sufficient in practice. We have used the Quasi-Newton line search algorithm as optimizer. The loss in variance reduction is at worse 1.8 dB compared to the direct method while the speed of computation is one hundred times faster for large sample sets. Examples of optimized sample sets are shown in Fig. 5.1. Observe that for $l = 0.1$ (Fig. 5.1(a)) , there are fewer samples at



a) Step at $\theta = \pi/8$

$\sigma^2 = 0$	$l = 0.1$	$l = 0.2$
$RMSE$	0.11	0.15
$IERR$	4.9%	21,2%



b) Step at $\theta = 3\pi/8$

$\sigma^2 = 0$	$l = 0.1$	$l = 0.2$
$RMSE$	0.30	0.17
$IERR$	12.4%	6.6%

Black line Original incident radiance $L(\theta, 0)$

Green line Mean value of incident radiance

Red line Radiance prediction with $l = 0.1$ for figures (a) to (d) and $l = 0.53$ for figures (e) and (f)

Blue line Radiance prediction with $l = 0.2$

RMSE Root mean squared error of the radiance prediction

IERR Relative error of the integral estimate

The integral value is 0.1464 in cases (a), (b), (e), (f) and 0.8536 in cases (c) and (d).

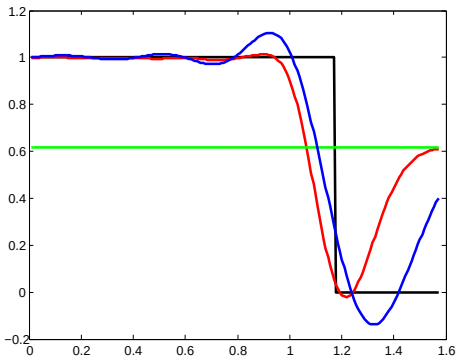
Figure 5.2: First Part. Behavior in case of a step radiance function

grazing angles since with this short lengthscale, sample points need to be closer to each other and the optimizer tends to trade off sampling directions with low weight (i.e. defined by $cos(\theta)$ weight function) for high weight sampling directions. This behavior is similar to importance sampling, however, in optimized sample sets, the effect of the cosine weight function tends to be counterbalanced by the covariance function as the lengthscale increases as shown in Figure 5.1(b).

5.5.5 Behavior in case of radiance function discontinuities

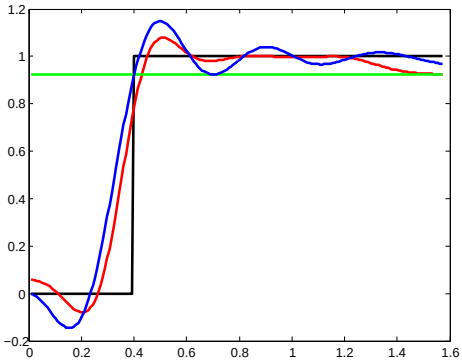
As discussed in section 5.4.2, the smoothness assumption for the integrand is modeled by the covariance function defined by Equation (5.2) and its strength is characterized by the lengthscale hyperparameter. Besides, as the power spectrum of a signal is obtained by taking the Fourier transform of its covariance function, we can also interpret the effect of lengthscale parameter in the frequency domain. Fixing a lengthscale value amounts to assuming that the bandwidth of the difference function $L(.)-\bar{L}(.)$ is limited. Recall however that this interpretation is valid only if the covariance function is stationary.

In this section, we analyze the behavior of the Bayesian Monte Carlo estimator of the irradiance integral when the radiance function is highly discontinuous. For this purpose, we have computed the BMC estimate of the irradiance integral of Equation (5.13) when the radi-



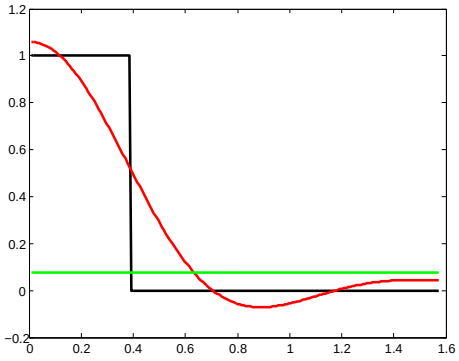
c) Step at $\theta = 3\pi/8$

$\sigma^2 = 0$	$l = 0.1$	$l = 0.2$
$RMSE$	0.28	0.16
$IERR$	2.1%	1.14%



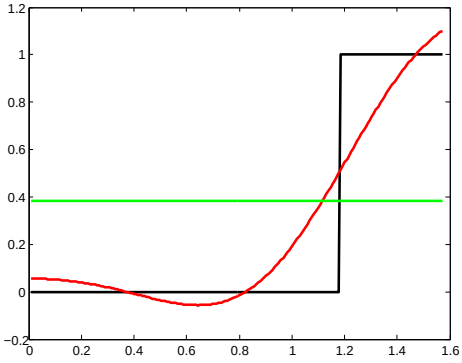
d) Step at $\theta = \pi/8$

$\sigma^2 = 0$	$l = 0.1$	$l = 0.2$
$RMSE$	0.13	0.18
$IERR$	0.8%	3.6%



e) Step at $\theta = \pi/8$

$\sigma^2 = 0.31$	$l = 0.53$
$RMSE$	0.16
$IERR$	2.1%



f) Step at $\theta = 3\pi/8$

$\sigma^2 = 0.31$	$l = 0.53$
$RMSE$	0.17
$IERR$	6.6%

Figure 5.2: Second Part. Behavior in case of a step radiance function

ance function is a step function in θ ($L(\theta,\phi) = s(\theta),\ \forall\,\phi$) and taking a constant mean function $\bar{L} = \frac{1}{2\pi}\int_{\Omega_{2\pi}} L(\theta,\phi)d\Omega$. We have also computed the predicted radiance function according to the regression formula of Equation (5.4).

The results presented in Figure 5.2 a) to d) use the two sample sets shown in Figure 5.1. We can observe that the $RMSE$ of the radiance prediction is generally high whereas the error on the integral estimate remains quite low except for some extreme cases. Moreover, when comparing case (a) and (c) for $l = 0.1$, we can see that the prediction $RMSE$ is much greater for case (c) than for case (d) whereas the order is reversed as regards $IERR$. This behavior is due to the particular sample points distribution of optimized sets. As discussed in section 5.5.4, the optimizing process tends to gradually allocate more points near the pole than near the equator and this is particularly well visible in Figure 5.1 when $l = 0.1$. Therefore, the prediction error is high at grazing angles but this has little effect on the integral estimate because of the weight function contribution. In cases (b) and (c), observe that when θ is close to $\pi/2$, the predicted radiance approaches the mean value because the sample points are too sparse and the effect of the covariance function becomes negligible in Equation (5.4).

Cases (a) and (d) are the most critical ones because the step occurs near the pole where the weight function has high values. Let us note that the longest lenghscale ($l = 0.2$) corresponds roughly to a geodesic distance of $\pi/8$, which is also the length of the step function of cases (a) and (d). In these cases, the $l = 0.1$ optimized design provides much better estimate because the smoothness assumption is less strong (i.e. the assumed signal bandwidth is higher) and also because the optimizer has allocated more sample points near the pole as explained in section 5.5.4. Let us note also that in both these cases, the choice of l has a much stronger impact on $IERR$ than on $RMSE$. In case (b) with $l = 0.1$, the sample points are too sparse at grazing angles to provide a good estimate.

Cases (e) and (f) show the results obtained with the optimal hyperparameter values (and associated optimal sample set) used for the rendering of the Sibernik Cathedral shown in Figure 5.15. The relatively high value of the optimal σ hyperparameter (i.e. the standard deviation of the additive noise ε) has the effect of smoothing out high frequency components as explained in [RW06], which reduces the ringings. The estimate of the irradiance integral remains good despite this strong smoothing effect and is even improved compared to case (a). As integration is inherently a low-pass filtering operation, smoothing out high frequency components does not affect too much the integral estimate. Another interpretation of this hyperparameter settings is to consider that the high frequency components of the original signal are included in the additive noise ε.

All these observations clearly show the importance of optimizing the design (sample points location, hyperparameters values and mean function) according to the integral estimate and not the radiance function prediction.

5.5.6 Discussion on implementation strategies

Basically, two implementation strategies are possible. The first one consists in adapting the sampling directions and the hyperparameters values at each irradiance integral evaluation so as to obtain the least possible variance and use the least possible number of samples. This solution is costly because it will require computing the quadrature coefficients at each point P,

which is $O(n^3)$ in complexity due to the inversion of the Q matrix.

Another possible strategy is to precompute sample sets that closely minimize the expected variance given by Equation (5.6). The quadrature coefficients associated with these sets can then be precomputed for the whole scene. In this case, a set of globally optimal hyperparameters must be selected at the scene level. This solution will be locally sub-optimal in terms of number of samples and hyperparameters values but it will avoid computing the quadrature coefficients at each integral evaluation. Another drawback of this solution is that it does not allow to take advantage of the freedom in samples distribution (recall that in Bayesian Monte Carlo, samples do not need to be drawn from a predefined distribution).

A combination of the two methods is possible. For example, we could take a precomputed optimal distribution and then supplement it with additional samples in direction of light sources. The obtained Bayesian Monte Carlo estimate remains unbiased as BMC is independent of the sample distribution. In the implementation proposed in this chapter, we have only used precomputed sample sets, other alternatives will be studied in the future.

5.5.7 Implementation details

Using Equation (5.5) and (5.15), the BMC estimate $\hat{E}_{BMC}$ of the irradiance value E at each visible point P can be written as:

$$\hat{E}_{BMC} = \bar{E} + \pi \, \mathbf{z}^t Q^{-1} (\mathbf{Y} - \bar{\mathbf{F}}) \tag{5.24}$$

where

$$\begin{aligned} \bar{E} &= \pi \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} \bar{f}(\eta) p(\eta) d\theta d\phi \\ \eta &= (\theta, \phi) \\ p(\eta) &= \frac{\sin(2\theta)}{2\pi} \end{aligned}$$

For a given set of directions over the hemisphere $\mathcal{D} = \{(\eta_i, Y_i) \,|\, i = 1, \ldots, n\}$, Y_i is the incoming radiance from direction η_i. This radiance is computed using a query to a photon map. $\bar{E}$ can be seen as an a priori estimate of the irradiance, deduced from the prior mean $\bar{f}$.

We have shown that Bayesian Monte Carlo integration depends on the GP model of the radiance function $L(\theta, \phi)$. In order to perform BMC integration, we need to determine the mean function $\bar{f}$ and the hyperparameters values of the GP model w_0, l and σ.

5.5.7.1 Determination of the mean function

As explained in section 5.4.3.1, Bayesian Monte Carlo allows us to infer $\bar{f}$ from the observed data $\mathcal{D}$. The choice of $\bar{f}$ impacts the variance of BMC estimate, but the effect of this additional term decreases rapidly as n increases and/or the value of l increases. When n is small or when the chosen value of l is not coherent with the data, the choice of $\bar{f}$ will have a large impact on $\hat{E}_{BMC}$. Ultimately, when $l = 0$, $\mathbf{z}$ is the null vector and Equation (5.24) gives:

$$\hat{E}_{BMC} = \bar{E} = \pi \int_{\Omega} \bar{f}(\eta) p(\eta) d\eta$$

which can be biased. As the determination of the hyperparameters may be costly and/or inaccurate, we want to choose $\bar{f}$ such that in extreme cases, the BMC estimate $\hat{E}_{BMC}$ is the same as the classical Monte Carlo (MC) estimate $\hat{E}_{MC}$ (computed from Equation (5.13)):

$$\hat{E}_{MC} = \frac{2\pi}{n} \sum f(\eta_i) \cos(\theta_i)$$

Note that when $\bar{f}$ is constant, $\bar{E} = \pi \bar{f}$. In order to have a simple worst-case unbiased BMC estimator, we take $\bar{f}$ as a constant function such that $\bar{f} = \hat{E}_{MC}/\pi$. When $l = 0$, we have:

$$\hat{E}_{BMC} = \bar{E} = \pi \bar{f} = \hat{E}_{MC}$$

Hence, in extremes cases, our estimate is equivalent to the classical Monte Carlo estimate. Note that the previous equations hold only for uniformly distributed samples. When using cosine importance sampling, the equation used to compute $\hat{E}_{MC}$ is the corresponding Monte Carlo estimator:

$$\hat{E}_{MC} = \frac{\pi}{n} \sum f(\eta_i)$$

To determine the impact of the choice of $\bar{f}$ on the Bayesian Monte Carlo estimate in the case of global illumination, we perform several tests. We choose a given point P in the scene and evaluate the irradiance incoming at P. Then we compute $\bar{f}$ as explained above (*MonteCarloMean*) and compare this method with the one using a classical arithmetic mean (*UniformMean*). We perform 2000 estimations of the incoming irradiance at P for each method. We then determine the bias and the mean quadrature error of these estimates, with respect to a reference value. The mean quadrature error of the Bayesian Monte Carlo estimates is compared with the mean quadrature error of a classical Monte Carlo estimator using the same number of rays.

Figure 5.3 shows the evolution of the bias of BMC estimator, depending on the values of l and the method used to compute $\bar{f}$. When $l = 0$, a bad choice of $\bar{f}$ (*UniformMean*) yields a biased estimate. Choosing $\bar{f}$ as explained above (*MonteCarloMean*) greatly reduces the bias. As Figure 5.4 shows, for small values of l, the mean quadrature error of the BMC estimate is also highly dependent on the method used to compute $\bar{f}$. However, we observe that for larger values of l, this dependency greatly diminishes and tends to zero. Recall that for a constant value of n, increasing l has the same effect that increasing n for a constant value of l. This means that the choice of $\bar{f}$ is significant when l or n are low. If we can generate a lot of samples, or if the signal has a wide covariance function (l is large), the impact of the choice of $\bar{f}$ on the bias and the mean quadratic error of the BMC estimate is almost negligible.

5.5.7.2 Determination of the hyperparameters

The covariance function $k(\mathbf{x}, \mathbf{x}')$ we chose is the square exponential function, parameterized by l and w_0. However, the quadrature coefficients $\mathbf{c} = Q^{-1}\mathbf{z}$ do not depend on the value of w_0^2 directly but on the ratio σ^2/w_0^2. Therefore, the result of Bayesian Monte Carlo integration depends only on the values of the hyperparameters l and $\sigma'^2 = \sigma^2/w_0^2$. In the following, we will consider that $w_0^2 = 1$ and $\sigma'^2 = \sigma^2$.

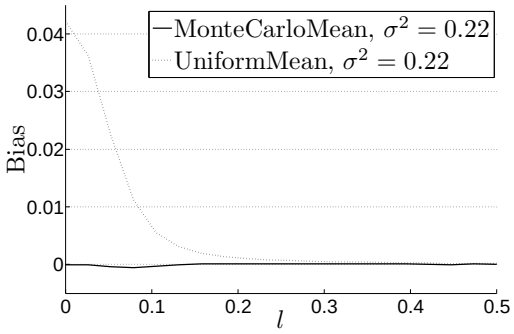


Figure 5.3: Depending on the value of l, the impact of $\bar{f}$ can be very important. For $l = 0$, a bad choice of $\bar{f}$ leads to a biased estimate. $n = 256$ samples.

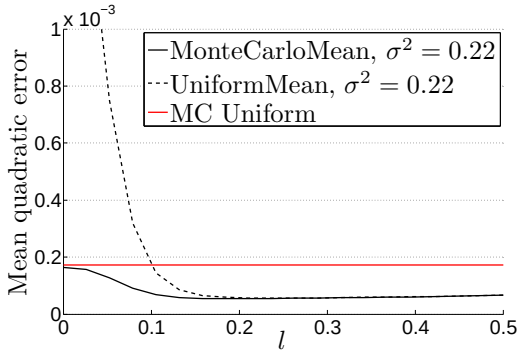


Figure 5.4: For small values of l, the mean quadratic error is highly dependent on $\bar{f}$. When l grows, the error converges to 0 whatever $\bar{f}$ is. $n = 256$ samples.

In order to determine the impact of the choice of l and σ^2 on the BMC estimate, we have studied the behavior of BMC integration for a given point P in a classical diffuse Cornell Box scene. We computed the irradiance E at point P using Monte Carlo (MC) and BMC integration. We performed 2000 integrations for each method and computed the variance of both estimators. Figure 5.5 shows the gain in quality obtained (in terms of variance reduction) with BMC integration compared to MC integration. Because of the the choice we made for $\bar{f}$, when $l = 0$ the variance of the BMC estimate is the same as that of the MC estimate. We can see that the optimal values for the hyperparameters are $l \approx 0.2$ and $\sigma^2 \approx 0.3$. Note that even with significant deviations from these optimum values, BMC performs better than MC. We did the same study for several visible points in the scene, and found that BMC integration outperforms MC integration. Of course, the optimal values obtained for l and σ^2 were different at each point.

As explained in section 5.4.3.3, finding the optimal hyperparameters that maximize the marginal likehood is costly. We prefer determining the hyperparameters that best fit the covariance function estimated at P from training data because this solution can be extended at a scene level as explained below. Figure 5.6 shows $\tilde{k}(d)$, the computed covariance function of the incoming radiance at point P. Using least-square fitting applied to the model given by Equation (5.20), we obtain $w_0^2 = 6.2\,10^{-3}$ and $l = 0.2615$. Since σ^2 is related to the divergence between the model and the actual data, we propose:

$$\sigma^2 \approx \frac{\int \left|(\tilde{k}(x) - k(x))\right| dx}{\int k(x)dx}$$

$k(x)$ being the model given by Equation (5.20), which yields $\sigma^2 = 0.31$ for the given example. Note that these values are close to the optimal values deduced from Figure 5.5.

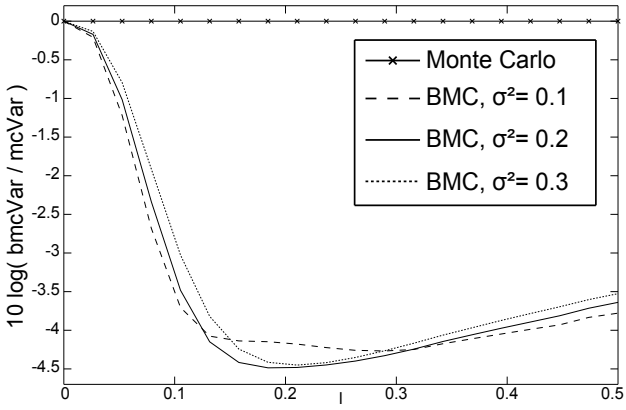


Figure 5.5: Gain in quality achieved by BMC for different values of l and σ^2, at a fixed point P within the Cornell Box scene ($n = 256$).

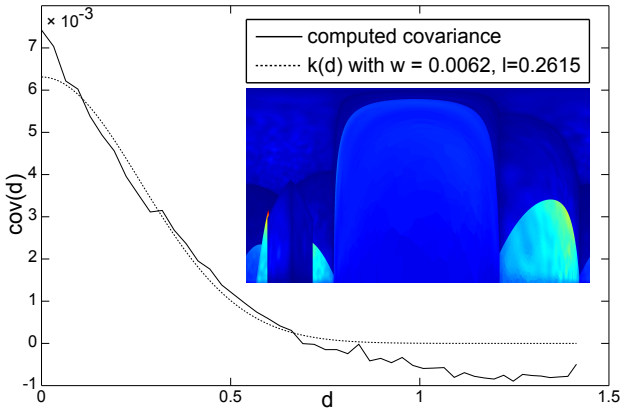


Figure 5.6: Covariance curve fitting and corresponding incoming radiance function in (ϕ, θ) coordinate space. 256k pairs of directions were generated to evaluate the covariance function.

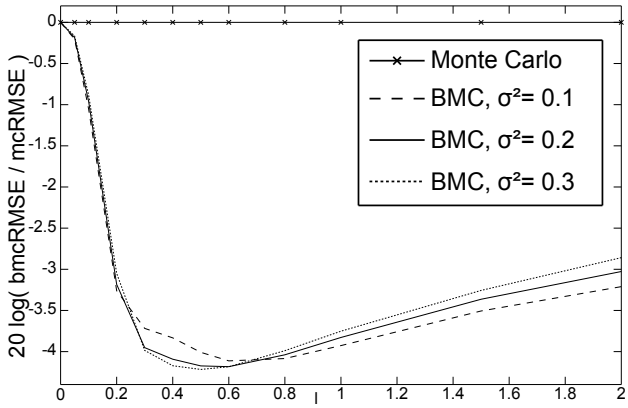


Figure 5.7: Gain in quality achieved by BMC with different values of $\bar{l}$ and $\overline{\sigma}^2$, for the rendering of a Cornell Box view ($n = 64$).

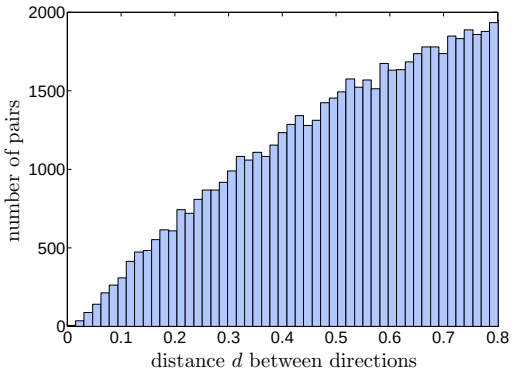


Figure 5.8: Histogram of the generated pairs, as a function of distance d. 228k pairs of uncorrelated directions have been generated but only few pairs range in $[0, \pi/4]$.

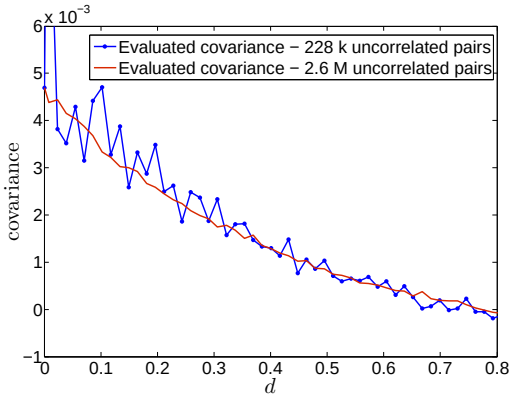


Figure 5.9: Covariance function $\tilde{k}(d)$ evaluated from pairs of uncorrelated directions. When using 228k pairs, $\tilde{k}(d)$ is noisy for small values of d. Using 2.6M pairs yields a better evaluation, but is very costly.

Evaluation of the covariance function In order to evaluate the covariance function $\tilde{k}(d)$ we cast N_c pairs of rays $(\eta_{1,i}, \eta_{2,i})_d$ from P, for each value of d. For a given d, $\tilde{k}(d)$ is given by:

$$\begin{aligned} \tilde{k}(d) &= \mathrm{E}\Big[(L(P,\eta_1) - \bar{L})(L(P,\eta_2) - \bar{L})\Big] \\ &\approx \frac{1}{N_c}\sum_{i=1}^{N_c}(L(P,\eta_{1,i}) - \bar{L})(L(P,\eta_{2,i}) - \bar{L}) \end{aligned} \tag{5.25}$$

where

$$d = \arccos(\eta_{1,i} \cdot \eta_{2,i}) \qquad \forall i \in [1, N_c]$$

$L(P, \eta_{1,i})$ is the radiance incoming at P from the direction $\eta_{1,i}$ and $\bar{L}$ the mean of all $L(P, \eta_{j,i})$.

To generate the pairs of rays $(\eta_{1,i}, \eta_{2,i})_d$, we have two solutions. The first solution consists in generating pairs of uncorrelated rays whose directions are independent and generated using a uniform distribution over the hemisphere. Then we classify these pairs with respect to their associated d. However, this classification poses some problems. Indeed, few pairs have a d close to 0 or to π, while most of the pairs have a d close to $\pi/2$ (Figure 5.8). This makes difficult to obtain good statistics when computing the associated covariance function using Equation (5.25). In addition, most of the time we are interested in the covariance function for the values of d ranging in $[0, \pi/4]$. Since there are very few pairs whose d is inside this range, the evaluated covariance function is very noisy (Figure 5.9). Even with rejection sampling (we reject all the pairs of directions whose d is above a threshold, without casting the associated rays) computing a good approximation of the covariance function requires a prohibitive amount of rays and is too costly. In addition, the classification process requires lot of memory.

The second solution consists in choosing one uniformly-distributed direction over the hemisphere, and then generating a second direction, whose distance d to the first one is equal to a given value. Let us suppose than we want to generate pairs to compute $\tilde{k}(d)$, where d is a

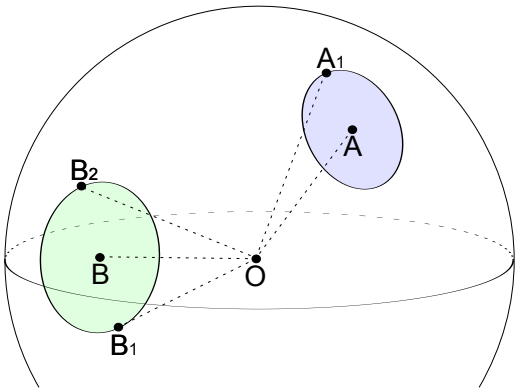


Figure 5.10: Directions A and B are generated on the sphere. Then a second direction is chosen for each of them, in the circle of radius d, lying on the sphere. As direction B_1 is lying under the upper hemisphere, it is rejected, and a third direction B_2 is draw.

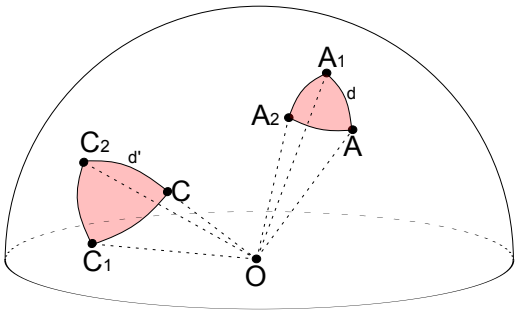


Figure 5.11: To reduce the number of rays shot during the hyperparameter determination step, we generate triplets of sample directions. Each sample direction A, A_1 and A_2 is at a distance d from the two other directions.

given distance. We choose a direction $\eta_{1,i}$, uniformly distributed over the hemisphere. Then, we choose $\eta_{2,i}$ such that the distance between $\eta_{1,i}$ and $\eta_{2,i}$ is d. This is done by computing the circle lying on the sphere, centered around $\eta_{1,i}$, which contains all the potential $\eta_{2,i}$. Then we uniformly choose a point in this circle. If we choose to consider the full circle, on the sphere, we will reject the pair if the chosen $\eta_{2,i}$ is lying beneath the upper hemisphere. If we consider only the part of the circle which lies on the superior hemisphere, $\eta_{1,i}$ is uniformly sampled in the remaining part of the circle (Figure 5.10).

These two solutions provide very similar results. Differences between these two solutions are hardly visible only for values of d greater than $\pi/4$. Consequently, the two solutions yield the same values of hyperparameters when using a square exponential model fitting. In addition the second method allows us to generate pairs of directions for given distances d, avoiding then a costly classification process. Since we can choose the number of pairs that can be generated for each d, we can perform a better approximation of $\tilde{k}(d)$ (Figure 5.6).

In our implementation, we chose the second solution (i.e. the directions of the rays inside a pair are correlated). To reduce the number of rays shot during the hyperparameters determination step, we generate triplets of sample directions. Each sample direction is at a distance d from the two other directions (Figure 5.11). We obtain 3 pairs of directions with only casting 3 rays, while 6 rays are needed in the classical way. We are aware that the process used to choose these pairs may introduce some error compared to the ground truth solution (the true covariance function). However, in practice, we have observed negligible differences in the computed hyperparameters. As shown above, a small deviation of the hyperparameters is not significant when using BMC.

To compute $\tilde{k}(d)$ associated with the incoming radiance at point P (both showed in Figure 5.6), we use a total number of 262k rays.

Global hyperparameters The number of additional rays is obviously too high if the computation explained in the previous section had to be performed for each new point P. To overcome this problem, we propose to determine a global value of l and σ^2 for the whole rendered image and use these global values $\bar{l}$ and $\bar{\sigma}^2$ in each final gathering BMC integration. Based on Equation (5.25), we compute a global covariance function $\bar{k}(d)$:

$$\bar{k}(d) \approx \frac{1}{N} \sum_{i=1}^{N_c} (L(P_i, \eta_{1,i}) - \bar{L})(L(P_i, \eta_{2,i}) - \bar{L})$$

where P_i is a visible point. For each value of d, we trace pairs of rays $(\eta_{1,i}, \eta_{2,i})$ from several different visible points P_i. After having fit the obtained $\bar{k}(d)$ values to Equation (5.20), we get $\bar{l}$ and $\bar{\sigma}^2$. For a given view of the Cornell Box scene, we obtain $\bar{l} = 0.45$ and $\bar{\sigma}^2 = 0.22$. Figure 5.7 shows the quality improvement (BMC RMSE over MC RMSE) obtained for this scene with different values of $\bar{l}$ and $\bar{\sigma}^2$ (RMSE has been computed with respect to a reference rendering using 16k final gathering rays per visible point). The values obtained with the fitting process are close to the optimal ones that we can deduce from Figure 5.7, say $\bar{l}_{fig} = 0.5$ and $\bar{\sigma}^2_{fig} = 0.3$. While these global values may be locally sub-optimal for a particular visible point, they prove to be quite good estimates. We used a total of 262k rays to estimate $\bar{k}(d)$ with $d \in [0, \pi/4]$: the cost of evaluating our hyperparameters is equivalent to the cost of one additional gathering ray per visible point.

5.5.7.3 Making BMC rendering practical

During rendering, we want to compute the BMC estimate given by Equation (5.24) for each visible point of the scene. As shown in section 5.5.7.1, evaluating $\bar{E}$ and $\bar{f}$ is straightforward. We still have to compute the values of $\mathbf{z}$ and Q^{-1}. Naively evaluating the integral for $\mathbf{z}$ and inverting Q for each visible point is not practical. In section 5.5 we showed that the values of $\mathbf{z}$ and Q depends only on the relative position of the sampling directions η_i within a given set of directions, $\mathcal{D}$. Thus, we propose to precompute the whole vector of quadrature coefficients $\mathbf{c} = Q^{-1}\mathbf{z}$ for a fixed number of sets $\mathcal{D}_j$, then use only these sets at rendering time. Since a set of sampling directions can be rotated around the normal without changing the quadrature coefficients (cf. 5.5.2), we have found that a small number N_d of different sets $\mathcal{D}_j$ is sufficient to reduce the artifacts due to repeated sampling patterns. In addition, N_d can be reduced as the number of samples n per set increases.

We have shown in section 5.5.2 that the values of $f_z(\eta_i)$ depend only on θ_i and l. In addition, θ_i always ranges between 0 and $\pi/2$, and the typical values of $\bar{l}$ are between 0 and 1. We chose to precompute $f_z(\theta_i, l)$ for a set of values of θ_i and l and store it into a table. At runtime, $f_z(\eta_i)$ is interpolated from the closest entry values in the table. Moreover, due to the properties of the covariance function $k(\mathbf{x}, \mathbf{x}')$, the matrix Q is symmetric and positive-definite. We can use Cholesky factorization to speed-up the matrix inversion process. The resulting algorithm is given by Algorithm 3. Note that the sets of directions $\mathcal{D}_j$ and their associated quadrature coefficient vector $\mathbf{c}_j$ can be computed on-the-fly before rendering, or read from a scene-independent database.

Algorithm 3: Final gathering algorithm using Bayesian Monte Carlo.

```
// Determination of the hyperparameters
Compute k̄(d) by casting pairs of rays;
Compute l̄ and σ̄² by fitting;
// Get N_d sets of directions D_j and the associated c_j
{(D_j), (c_j)} = GetSets (N_d, n, l̄, σ̄²);
// at rendering time, F.G. pass
foreach visible point p do
    Pick a set of directions D_j and associated c_j;
    Rotate D_j along the normal at p by a random φ;
    Cast rays according to D_j and compute Y;
    Compute f̄ and Ē_MC;
    // BMC estimate of the irradiance at p
    Ē_BMC = Ē_MC + π c_j (Y − f̄);
endfch
```

5.5.7.4 Optimal sampling

Basically, Bayesian Monte Carlo integration schemes do not rely on random samples. In our application to final gathering, instead of generating $\mathcal{D}_j$ using random directions, we could search for a set of directions which minimizes Equation (5.7) as explained in section 5.5.4. Although the optimization process is computationally expensive, the generated optimal sets $\mathcal{D}_j$ depend only on the values of N, $\bar{l}$ and $\bar{\sigma}^2$, but they do not depend on the scene. One could precompute a scene-independent database of optimal sets $\mathcal{D}_j$ for some values of n, $\bar{l}$ and $\bar{\sigma}^2$, storing directions and associated optimal $\mathbf{c}_j$. Then, during the rendering process, we only have to pick some of the optimal sets, trace corresponding rays and compute a dot product between $\mathbf{c}_j$ and $\mathbf{Y}$.

5.6 Future Works

Our current implementation does not fully exploit the advantages of the Bayesian Monte Carlo method. This is why we are planning to investigate several aspects of the original BMC method to adapt them to global illumination. First we would like to find a way of locally determining the hyperparameters. Second, to fit more precisely the discontinuities of the radiance function, we think it would be better to use Matérn class covariance functions. Finally extensions to glossy surfaces and path tracing as well as adaptive sampling would make the BMC approach more beneficial for global illumination in general.

5.6.1 Automatic local hyperparameters determination

As explained in Section 5.4.3.3, the Bayesian Monte Carlo method is based on a GP prior and assumes that the hyperparameters of the GP model are known. Our current implementation determines global hyperparameters before rendering. They fit the covariance function

Function `GetSets(`N_d, n, l, σ^2`)`

```
// Precompute N_d sets D_j of n directions η_i and their associated
   quadrature coefficient c_j
if Optimal sampling then
   Get optimal {(D_j), (c_j)} from a precomputed database of optimal sets;
else
   for j = 0 to N_d-1 do
      D_j = set of n random directions over the hemisphere;
      // compute z_j
      foreach direction η_i do
         Interpolate f_z(η_i) from the table;
      endfch
      Compute Q_j;
      Invert Q_j using Cholesky factorization;
      c_j = Q^-1 z_j;
   endfor
endif
```

estimated by casting rays towards the scene from the visible points (Section 5.5.7.2). These global hyperparameters are then used for all the computations performed during the rendering pass (where incoming irradiance is computed for each visible point). However, we think that locally determined hyperparameters would provide better results (Section 5.7.3 and Figure 5.15). In this way, we would determine a different set of optimal hyperparameters for each visible point.

One way to do this is to find $\hat{\vartheta}$, the set of hyperparameters which maximizes the likehood function (Section 5.4.3.3). Rasmussen [RW06] expresses the logarithm of the likehood function as:

$$\log p(\mathcal{D}|\vartheta) = -\frac{1}{2}\mathbf{Y}^t Q^{-1}\mathbf{Y} - \log|Q| - \frac{n}{2}\log 2\pi$$

For a given point P and a chosen set of directions $\mathcal{D}_i$, computing $\hat{\vartheta}$ is expensive. The likehood function needs to be computed several times, for different values of l and σ (because Q depends on them). This computation requires to invert several different matrices Q, which is not practicable. However, the log likehood function could be evaluated for a finite set of pairs (l, σ). Then we would choose the pair which yields the maximum log likehood value, and use it as an approximation of $\hat{\vartheta}$. This idea could be implemented as follows.

Let us take a given point P, choose a set of directions $\mathcal{D}_i$ and compute the associated $\mathbf{Y}_i$. Recall that in our current implementation, given global values for l and σ, we precompute a vector of quadrature coefficients for each $\mathcal{D}_i$, which is then used to evaluate the reflected radiance at P. Rather we could precompute quadrature coefficients vectors $c_{i,l,\sigma}$ for different values of l and σ. In addition, we would store the matrix $L_{i,l,\sigma}$, given by the Cholesky decomposition of the matrix $Q^{-1}_{i,l,\sigma}$ which was used to compute each vector $c_{i,l,\sigma}$. When computing the reflected

radiance at P, we evaluate the following equation for several values of (l, σ):

$$\mathbf{Y}_i^t Q_{i,l,\sigma}^{-1} \mathbf{Y}_i = \mathbf{Y}_i^t L_{i,l,\sigma} L_{i,l,\sigma}^t \mathbf{Y}_i = \mathbf{Y}_i^t L_{i,l,\sigma} \cdot \mathbf{Y}_i^t L_{i,l,\sigma}$$

As L is a triangular matrix, the evaluation cost is low. We then pick the pair (l, σ) which maximizes the log likehood function, and use the corresponding quadrature coefficients vector $c_{i,l,\sigma}$.

The computation cost of this approach would depend on the number of pairs (l, σ) for which we precompute $c_{i,l,\sigma}$ and $L_{i,l,\sigma}$, for each $\mathcal{D}_i$. In order to speed-up the selection of the best pair (l, σ) for a given $\mathbf{Y}$, we could also determine some heuristics which allow us to quickly eliminate some candidates. Finally, it is very likely that the optimal values of l and σ do not change significantly over a given surface. Consequently, we could employ a caching approach by computing an accurate approximation of $\hat{\vartheta}$ only for certain points, then using it for neighbor points.

We believe that automatic determination of the locally optimal hyperparameters would yield a large improvement.

5.6.2 Glossy surfaces

Our current implementation of Bayesian Monte Carlo considers only diffuse surfaces. The glossy reflections are computed using classical importance Monte Carlo. BMC could be interesting for computing a fast approximation of glossy reflections with few uniformly distributed rays. Indeed, recall that BMC reconstructs an approximation of the function $f(x)$ (regression), given a set of data (Equation 5.4). The BMC integration yields the exact integral of the product of the regression function and the function $p(x)$. Consequently, even with a sparse sampling, BMC could provide a smooth (*i.e.* less noisy) approximation of glossy reflections. Of course, Bayesian Monte Carlo used in conjunction with importance sampling would yield better results.

Computing glossy reflections with Bayesian Monte Carlo should be straightforward. The only things which would change are the sampling strategy and the function $p(x)$. However, to make Bayesian Monte Carlo practical, we had to resort to a precomputation step. During this step, we chose to precompute the quadrature coefficients vector for a finite number of sets of sampling directions. For diffuse reflections, the sample direction sets depend only on the number of rays used for final gathering. However, when considering glossy reflections, we may want to generate different sets of directions for each shininess value.

Another aspect of the problem is the computation of the vector $\mathbf{z}$. For diffuse reflections, the function f_z depends only on two parameters, l and θ_i (Section 5.5.3), and $\mathbf{z}$ can be efficiently precomputed. Indeed, we use a 2-dimensional lookup table to quickly evaluate f_z for any sampling direction at runtime. For glossy reflections, the function f_z becomes 5-dimensional. The precomputation step becomes more complicated, and evaluating f_z at runtime for a given direction gets very difficult. It is not practicable anymore to precompute values of the 5-dimensional f_z function and use a lookup table. However, we noticed that the function f_z is relatively smooth and could be approximated by polynomial functions which would be evaluated at runtime efficiently.

5.6.3 Adaptive sampling

Most Monte Carlo algorithms allow adaptive sampling. The decision of taking additional samples is generally based on the variance of the already obtained samples. If the variance is higher than a given threshold, more samples are taken. Usually, the additional computation needed to compute the new estimate from the new sample values is low.

For example, let us evaluate the irradiance at a given point P, using classical importance Monte Carlo, with cosine-distributed directions over the hemisphere. We first compute an estimate E_n of the irradiance at P using n samples. As we want to improve, we cast m additional rays from P, with the same distribution. The new estimate E_{n+m} is then:

$$E_{n+m} = \frac{1}{n+m}(nE_n + mE_m)$$

where $L(\eta_i)$ is the incoming radiance along the direction η_i and E_m is:

$$\frac{1}{m}\sum_{i=1}^{m} L(\eta_i)$$

Note that the evaluation of the new estimate E_{n+m} requires only few basic operations, but a straightforward application of this method does not allow to globally optimize the sample distribution. The convergence of this method is not better than that obtained by sampling $(n+m)$ directions in one go and using importance Monte Carlo. However, E_n and E_m could be computed one after the other or in parallel.

Let us now consider Bayesian Monte Carlo. To evaluate the estimate E_n we have to invert a matrix Q of size $n \times n$. To take into account m new samples, we have to build a new matrix Q', from the $n+m$ samples. We then invert the matrix Q' which is $(n+m)\times(n+m)$ and then compute E_{n+m}. There is no other way to compute the BMC estimate E_{n+m} from E_n and E_m. In addition, if we only have the estimates E_n and E_m, without knowing the positions and the values associated with the additional samples, it is not possible to retrieve E_{n+m}. Indeed, we need the whole set of samples at once to be able to take into account the correlation between all the samples. This is similar to combining two estimates which have been computed using stratified Monte Carlo. The resulting estimate generally has a variance higher than that computed with the same total number of stratified samples used at once.

However, progressivity could be integrated during the preprocessed step. Instead of pre-computing samples sets composed of n samples, we could build a hierarchy of sets. These sets would be built by iteratively adding sample directions, then computing the associated quadrature vector, and so on. At rendering time, the appropriate level of the hierarchy is used.

5.6.4 Application to path-tracing

The application of Bayesian Monte Carlo to final gathering only takes into account the directional covariance properties of the incoming radiance. Indeed, the correlation between two sample values depends only on the angular distance between their associated directions. We would like to extend this approach to path-tracing. In this case, we would consider a higher dimension integrand and a larger number of hyperparameters.

More precisely, to compute the final color of a pixel, we cast many rays through this pixel. Each ray R_i intersects the scene at a point $P_{i,0}$, then it is reflected in a direction $\eta_{i,0}$, intersects again the scene at another point $P_{i,1}$ and so on. Each light path is represented by the intersection points $P_{i,j}$ with $j = 0 \cdots n$, n being the length of the light path. The covariance between the radiance incoming from two different light paths could be defined as a function of the distance between the respective intersection points and the angular distances of the respective reflection angles along the paths.

In our current implementation, we apply BMC to final gathering. This is equivalent to considering that all the $P_{i,0}$ are the same, and only taking into account the angular distance between the $\eta_{i,0}$. We do not consider subsequent bounces. Consequently, the correlation between the values associated with two independent sample values (radiances) S_i and S'_i is expressed by $cov(S_i, S'_i) = k(\eta_{i,0}, \eta'_{i,0})$, where k is a chosen covariance function.

However, when considering path tracing, the covariance function needs to take into account the positions of the different intersection points. Depending on the number of bounces we consider, our covariance function can be expressed as:

$$\begin{aligned} cov(S_i, S_{i'}) &= k'(P_{i,0}, \cdots, P_{i,n}, P'_{i,0}, \cdots, P'_{i,n}, \eta_{i,0}, \cdots, \eta_{i,n}, \eta'_{i,0}, \cdots, \eta'_{i,n}) \\ &= \prod_{m=0}^{m=n} kp_m(P_{i,m}, P'_{i,m}) \prod_{m=0}^{m=n} k\eta_m(\eta_{i,m}, \eta'_{i,m}) \end{aligned}$$

where k' is a separable function. kp_m and $k\eta_m$ could be square exponential functions, parameterized by the spatial hyperparameters lp_m and the angular hyperparameters $l\eta_m$. It is very likely that the spatial and angular hyperparameters will be very small for values of m higher than 0. Indeed, the correlation between two samples mainly depends on the proximity of the first intersection points $P_{i,0}$ and $P'_{i,0}$ and the angular difference between the first reflection directions.

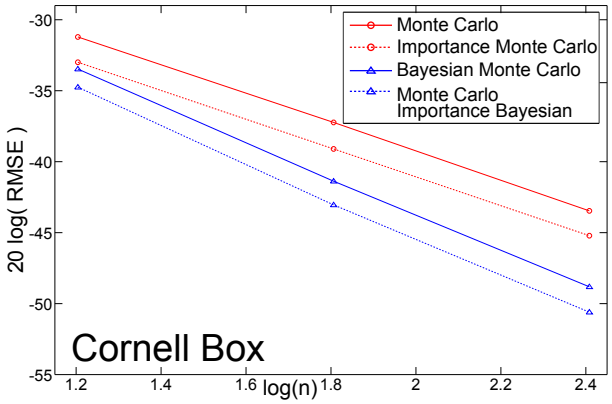
Compared to our implementation, the extension of Bayesian Monte Carlo to path tracing would require a covariance function which depends on one additional parameter, lp_0.

5.7 Results

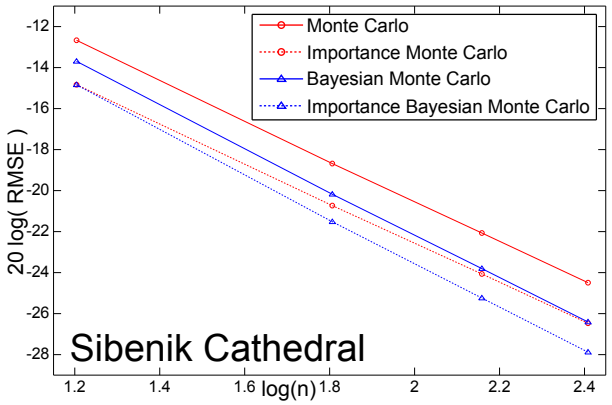
All results were gathered on a Intel Core Duo T2600 (2.16GHz) with 2GB RAM.

5.7.1 General observations

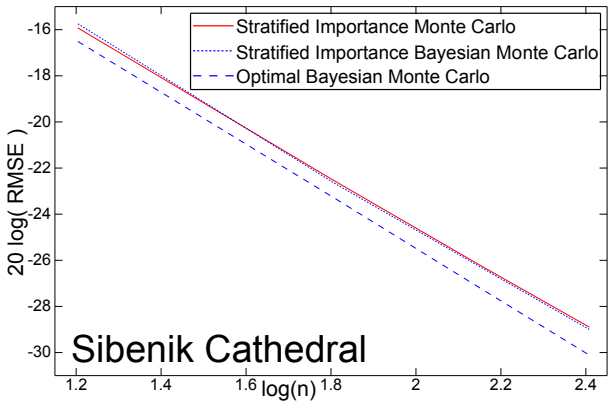
During the final gathering pass, our algorithm does not have to draw random numbers to generate directions on the hemisphere. Instead, we pick a previously computed set $\mathcal{D}_j$ of directions and retrieve the associated quadrature coefficient vector $\mathbf{c_j}$. Then, the only additional computation with respect to MC integration is a dot product between $\mathbf{c_j}$ and $\mathbf{Y}$. During the final gathering pass, the computation time of our algorithm is essentially the same as that of classical Monte Carlo (the computation time is dominated by ray casting and photon map queries). So, the overhead of our algorithm lies in the evaluation of the hyperparameters and the computation of the sets $\mathcal{D}_j$ together with the associated quadrature coefficient vectors.



(a)



(b)



(c)

Figure 5.12: RMSE variation as a function of the number of rays. *(a)* Cornell Box scene. *(b)* and *(c)* Sibenik Cathedral scene. Compared to SI-MC, SI-BMC gives similar results. However, Optimal BMC significantly improves the rendering quality.

We found that we can get a good approximation of the actual covariance function of the incoming radiance using a total number of $N_c = 262k$ rays. In our test scenes, the determination of $\bar{l}$ and $\bar{\sigma}^2$ takes 1.1 s for the Cornell Box scene, 5 s for the Sibenik Cathedral scene and 7s for the Sponza Lucy scene.

On-the-fly generation of sets of directions and the associated quadrature coefficients depends mainly on n, the number of rays shot from each visible point. The cost of this generation is dominated by the inversion of the $n \times n$ matrix Q. We found that we do not need a large number N_d of sets $\mathcal{D}_j$, to avoid artifacts. Moreover, the larger the number of samples n, the smaller N_d. Computing all the $\mathbf{c}_j$ takes 273 ms when $n = 16$ and $N_d = 1000$, 1.16 s when $n = 64$ and $N_d = 250$ and 9.2 s when $n = 256$ and $N_d = 60$.

5.7.2 Bayesian Monte Carlo integration using random sampling

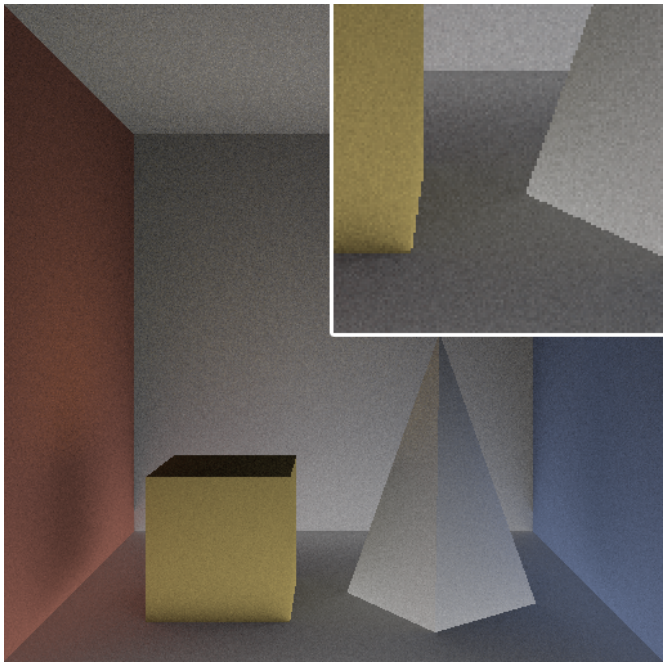
As Bayesian Monte Carlo does not make any assumption on the sample directions η_i, we can make use of importance sampling and/or stratified sampling to generate sample direction sets $\mathcal{D}_j$. In this way, we will be able to compare BMC with MC for the same sampling directions $\mathcal{D}_j$ generated either with uniform sampling or importance sampling.

Figure 5.12 compares the variation of RMSE with respect to a reference rendering, as the number of samples increases. We can see that for the exact same $\mathcal{D}_j$, BMC greatly reduces the noise in the rendered image (Cornell Box, Figure 5.13 and 5.12(a)). We can notice that BMC (with importance sampling) outperforms MC (with importance sampling) (Sibenik Cathedral, Figure 5.12(b)). Table 5.1 gives some results on timing and image quality obtained with BMC and MC from the same sets of sample directions $\mathcal{D}_j$.

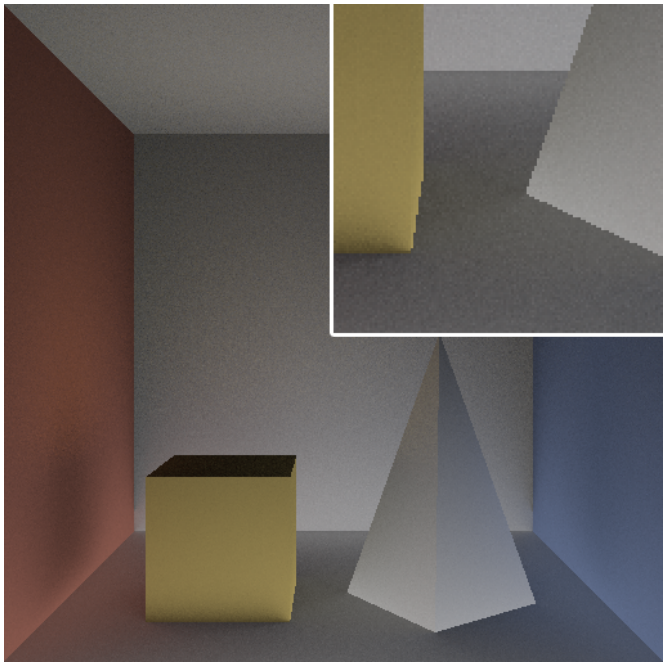
Figure 5.14 shows the rendering of a scene containing glossy objects. Our current implementation of BMC computes final gathering only on diffuse surfaces. The glossy reflections were computed using Monte Carlo with importance sampling. When comparing RMSE values of different methods, we only consider the pixels computed with BMC. Figure 5.14(e) and (f) show the difference images. We can see from these difference images that the error is generally smaller with BMC especially for bright surfaces but this is hardly visible on the rendered picture. This is due to the fact that, thanks to the Gaussian Process prior of BMC, a few samples are necessary to evaluate the contribution of a bright surface seen from a relatively narrow angle whereas much more samples are necessary with importance sampling.

5.7.3 BMC estimator using optimal sets of directions

BMC has the advantage (over MC) of assigning small weights to nearby samples and larger ones to relatively distant samples. As for SI-MC (Monte Carlo with stratified importance sampling), nearby samples are rare, only relatively distant samples are generated. This is why SI-BMC (BMC with stratified importance sampling) does not perform better than SI-MC, but provides results that are similar to those of SI-MC in the worst case. However, with O-BMC (BMC with optimal sample sets $\mathcal{D}_j$), the obtained results are better as the method significantly reduces noise in the rendered image. Note that if we want SI-MC to achieve the same RMSE as the ones of O-BMC, more than 33% additional sample rays are needed (we save 25% rays with respect to SI-MC, Table 5.1).



(a) Classical Monte Carlo



(b) Bayesian Monte Carlo

Figure 5.13: Cornell Box rendering (Indirect only). $n = 256$. To obtain the same quality, Monte Carlo integration would require more than 900 rays.

In Figure 5.15, the image quality is the same for SI-MC and O-BMC in some parts of the image (the global $\bar{l}$ and $\bar{\sigma}^2$ are far from optimal) and better for O-BMC in some other parts (the global $\bar{l}$ and $\bar{\sigma}^2$ are closer to optimal). This means that finding a fast method of computing local estimates of l and σ^2 would further improve the performances of O-BMC.

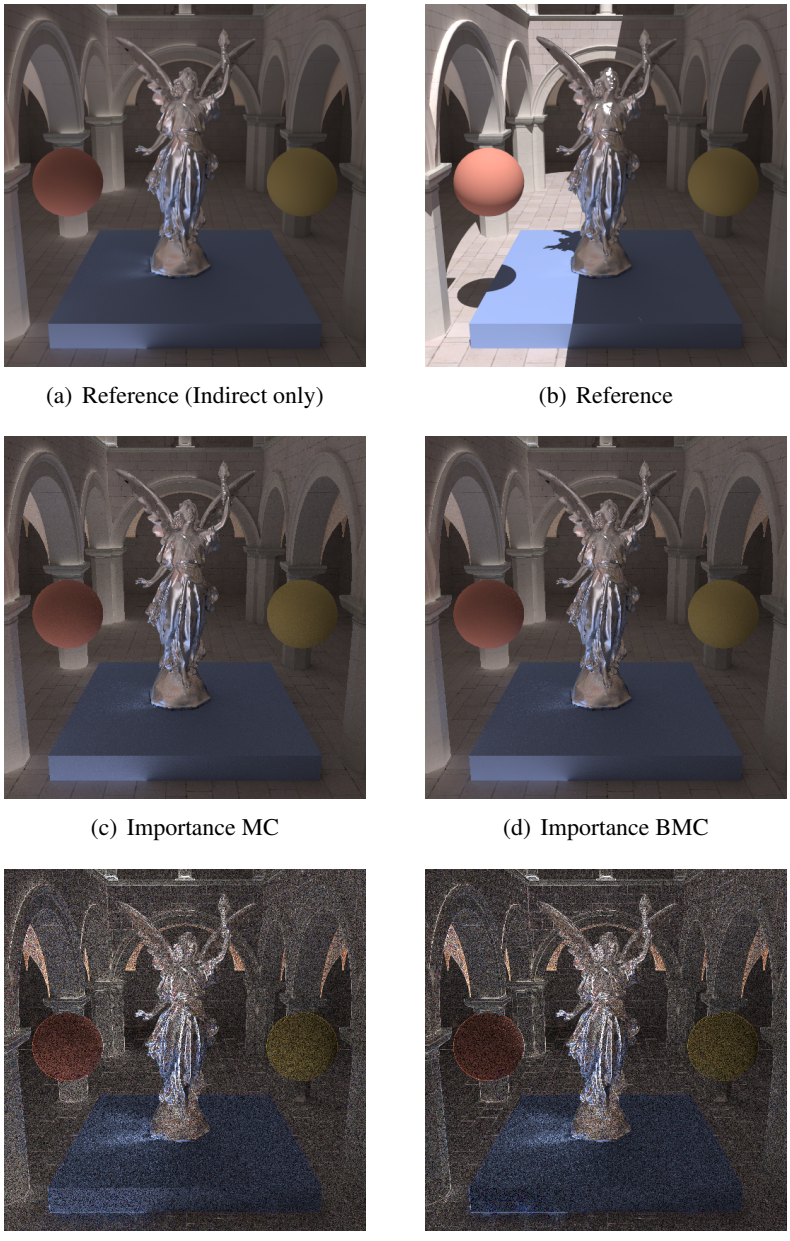
5.7.4 Discussion

The results that we have obtained are surprisingly good given the strong smoothness assumption that we have put in the Gaussian Process prior. First, we have used a very smooth covariance function as explained in section 5.4.2, second we have taken a constant mean function that does not allow to reduce radiance function discontinuities as discussed in section 5.4.3.1 and third, the model hyperparameters are selected at a global level, which leads to quite long lengthscale values, (i.e. narrow bandwidth assumption for the radiance function). These assumptions may seem incompatible with the sharp discontinuities met in the test scenes. However, if we consider the results presented in Figure 5.2, the integral estimate error remains below 7% with the longest lengthscale ($l = 0.53$). Furthermore, Figure 5.5 and 5.7 show that the variance of BMC estimates increases slowly with l when l is greater than the optimal value. These observations explain why optimizing hyperparameters at the scene level leads to rather long lengthscales since the error on the integrals estimates will remain low despite significant deviations from the optimal values.

Figure 5.16 shows the strong smoothing effect brought by the Bayesian prediction with the globally optimal hyperparameters, which strengthens the conclusions drawn from the simulations of section 5.5.5. Hyperparameters must be selected so as to obtain the best integral estimates and not the best data fittings. In this regard, our hyperparameters determination method leads to values that are close to optimum as explained in section 5.5.7.2. However, the simulation results presented in section 5.5.5 suggest that better estimates can be expected if the hyperparameters could be locally adapted.

5.8 Conclusion

This work clearly shows the benefits of Bayesian Monte Carlo over traditional Monte Carlo methods. Similarly to deterministic quadrature methods, BMC takes into account sample positions but remains a probabilistic method in that it assumes a Gaussian prior on the function to be integrated. Both these features contribute to the strength of this method. In the BMC implementation described in this chapter, we have chosen to leave unchanged the computing load of the rendering loop, i.e. estimates are computed by simple weighted sums just as in standard MC. We have shown that this method can lead to significant savings in terms of number of rays. However, as already mentioned above, this implementation does not fully exploit the advantages of the BMC method. Ideally, the prior model should be adapted to the data for each integral computation, which is too costly with the usual likehood minimization methods. Our future research will thus be directed toward new methods allowing local adaptation of the hyperparameters values as well as the mean function and the number of sample points. The results we have obtained with this first implementation lead us to think that there is a great



(a) Reference (Indirect only)

(b) Reference

(c) Importance MC

(d) Importance BMC

(e) MC-I (difference)

(f) BMC-I (difference)

Figure 5.14: Rendering of Sponza Atrium scene, with a glossy Lucy. $n = 256$, $\bar{l} = 0.42$, $\bar{\sigma}^2 = 0.31$. Glossy reflections have been computed using importance sampling. Difference pictures have been multiplied by 10. To obtain the same quality, Monte Carlo integration would require $n = 445$.

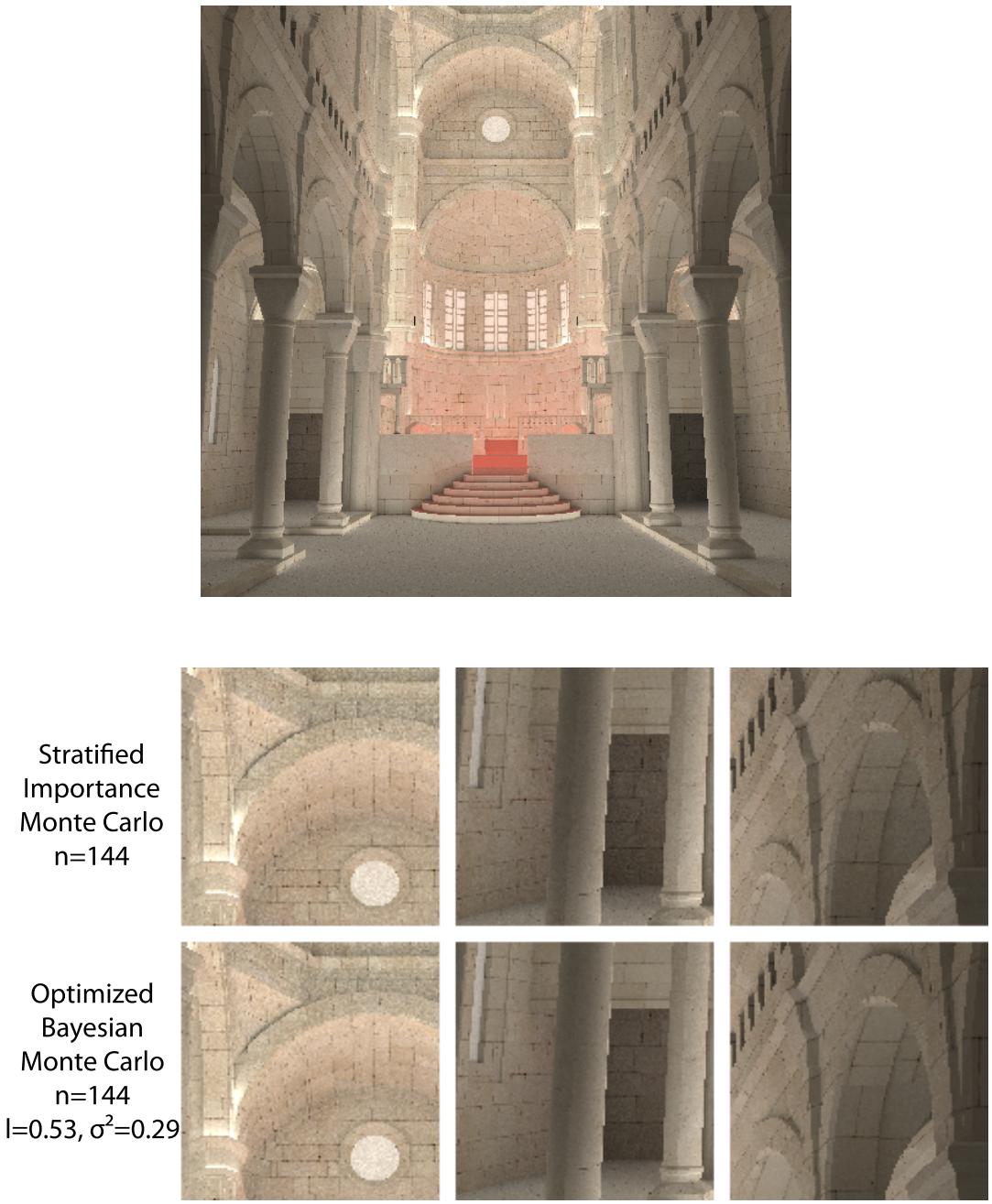
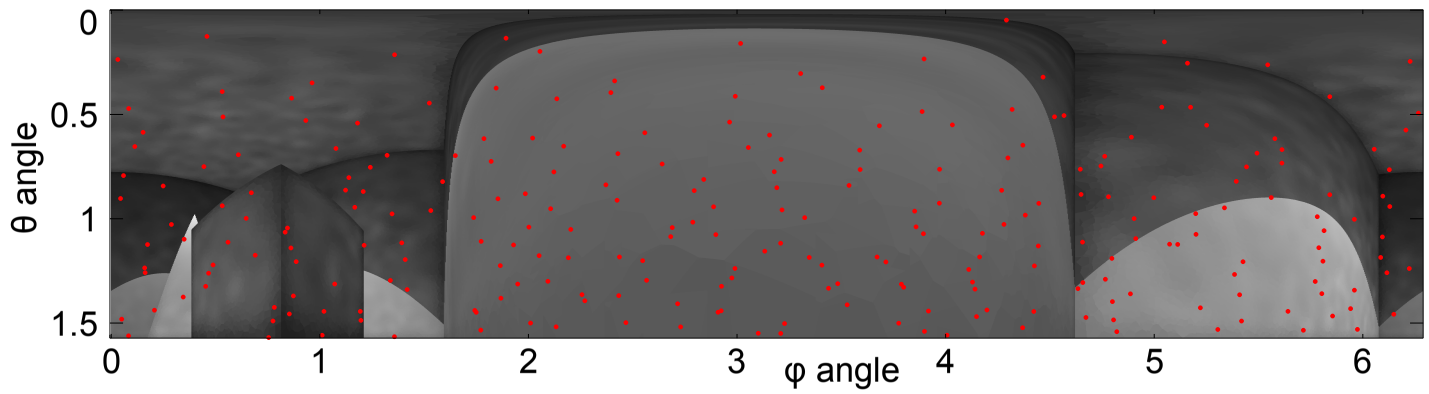


Figure 5.15: Optimal BMC rendering of the Sibenik Cathedral (Indirect only). $n = 144$, $\bar{l} = 0.53$, $\bar{\sigma}^2 = 0.29$. Locally, OBMC performance depends on how close are the global values of $\bar{l}$ and $\bar{\sigma}^2$ to the local optimal values. Note that O-BMC never performs worse than SI-MC, hence the global quality of the rendering is improved. The overall reduction of the RMSE value is 11%. The reduction of the RMSE value for each of the detail view is respectively 8%, 22% and 16%.

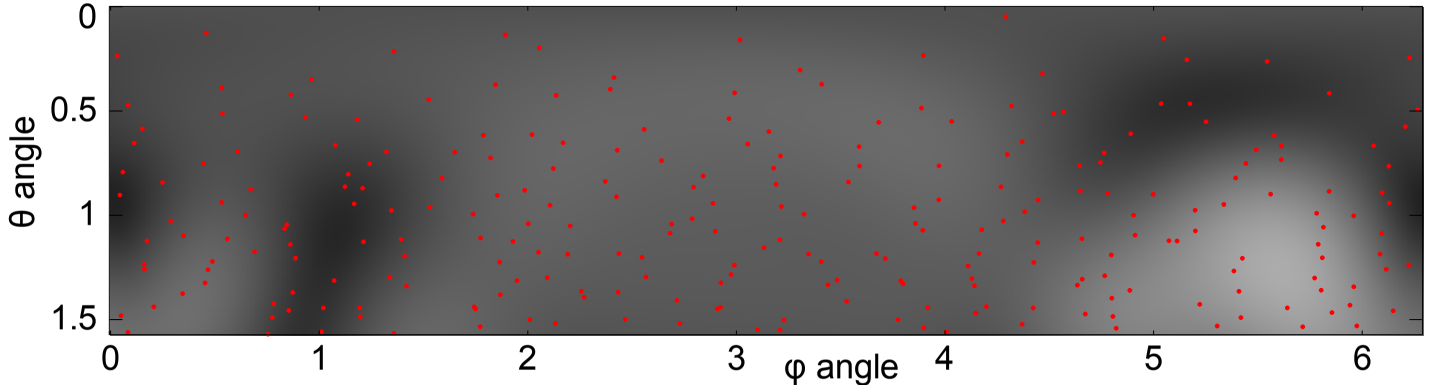
Scene	Sampling Method	# FG Rays	RMSE (10^{-3}) MC	RMSE (10^{-3}) BMC	RMSE Gain	Same quality MC #Rays	% Rays saved	Total time
Cornell Box	Uniform Sampling	256	6.71	3.63	5.34 dB	900	72%	211s
	Importance Sampling	256	5.49	2.95	5.40 dB	896	71%	
Cathedral	Uniform Sampling	256	59.6	47.8	1.92 dB	402	36%	1310s
	Importance Sampling	256	47.5	40.4	1.41 dB	362	29%	
	Stratified + Importance	256	35.9	35.5	0.1 dB	266	4%	1310s
	Optimized Sampling		/	31.2	1.22 dB	342	25%	1301s
Sponza Lucy	Importance Sampling	256	9.1	8.03	0.99 dB	445	49%	1648s
	Stratified + Importance	256	7.67	7.61	0.07 dB	271	6%	1648s
	Optimized Sampling		/	7.352	0.37 dB	332	23%	1639s

Table 5.1: Quality comparison between BMC renderings and the associated MC renderings (using the exact same sets $\mathcal{D}_j$). Total time includes determination of hyperparameters (few seconds, depending on the scene) and on-the-fly computation of $\mathcal{D}_i$ and associated $\mathbf{c}_j$ (9.2 sec when $n = 256$). The overhead of our algorithm is negligible with respect to the time spent on ray casting. *Same quality MC #rays* is the number of rays needed to compute a rendering yielding the same RMSE value as the corresponding BMC rendering.

margin for improvement in this direction. We are also planning to extend our BMC approach to the rendering of glossy surfaces.



(a) Incident radiance function



(b) Smoothing effect of the Bayesian prediction

Figure 5.16: Bayesian prediction of incident radiance seen from a given point in the Sibenik Cathedral scene. The red dots are the sample points.

Chapter 6

Conclusion

Our work focused on fast computation of global illumination and rendering based on Monte Carlo algorithms. We proposed two approaches: Photon-Driven Irradiance Cache and Bayesian Monte Carlo. In this chapter we first summarize them, then propose directions for future works.

6.1 Contributions

6.1.1 Photon-Driven Irradiance Cache

We have described a global illumination method combining two well known techniques: photon mapping and irradiance caching. The photon mapping method has the advantage of being view independent but requires a costly additional rendering pass, called final gathering. As for irradiance caching, it is view-dependent, irradiance is only computed and cached on surfaces of the scene as viewed by a single camera. To compute records covering the entire scene, the irradiance caching method has to be run for many cameras, which takes a long time and is a tedious task since the user has to place the needed cameras manually. Our method exploits the advantages of these two methods and avoids any intervention of the user. It computes a refined, view-independent irradiance cache from a photon map. The global illumination solution is then rendered interactively using radiance cache splatting.

6.1.2 Bayesian Monte Carlo for global illumination

We investigated a new approach to variance reduction for Monte Carlo rendering algorithms. Indeed, these algorithms typically rely on importance sampling to reduce the variance of the estimates. However, importance sampling is efficient when the proposal sample distribution is well-suited to the form of the integrand but fails otherwise. The main reason is that the sample locations are ignored. All sample values are given the same importance regardless of their proximity to one another. Two samples falling in a similar location will have equal importance whereas they are likely to contain redundant information. The Bayesian approach we have proposed in this paper uses both the location and value of the data to infer an integral value based on a prior model of the integrand. The Bayesian estimate depends only on the sample values and their given locations and not how these samples have been chosen. We have

shown how this theory can be applied to the final gathering problem and have presented results that demonstrate the benefits of Bayesian Monte Carlo.

6.2 Future works

6.2.1 Photon-Driven Irradiance Cache

6.2.1.1 Glossy surfaces

Our current Photon-Driven Irradiance Cache algorithm focuses on computing the indirect diffuse component of the global illumination solution. Although it takes into account the L(D|G)*DE paths (through the use of the photon mapping), we do not store the incoming lighting on glossy surfaces. We could extend our algorithm by generating a *radiance* cache on glossy surfaces too. A coarse approximation of this radiance cache would be quickly derived from the photon map, whereas a more accurate version of the cache, with gradients, could be computed in a following pass. However, computing glossy reflections only from a photon map, without resorting to a prohibitive number of photons, is still a subject of research.

6.2.1.2 Dynamic scenes

Our current implementation only considers static scenes. Some research work have already been made to adapt the radiance cache to dynamic scenes [GBP06]. Likewise, for photon mapping, interactive construction and visualization of the photon map have been proposed [FD09]. These methods could be combined to yield an algorithm which builds a temporal radiance cache over the scene, from an interactive photon map. The resulting method could be used for interactive walkthrough in dynamic scenes.

6.2.2 Bayesian Monte Carlo for global illumination

Our current implementation does not fully exploit the advantages of the Bayesian Monte Carlo method. This is why we are planning to investigate several aspects of the original BMC method to adapt them to global illumination. First we would like to find a way of locally determining the hyperparameters. Second, to fit more precisely the discontinuities of the radiance function, we think it would be better to use Matérn class covariance functions. Finally extensions to glossy surfaces and path tracing as well as adaptive sampling would make the BMC approach more beneficial for global illumination in general.

6.2.2.1 Automatic local hyperparameters determination

The Bayesian Monte Carlo method is based on a GP prior and usually assumes that the hyperparameters of the GP model are known. Our current implementation determines global hyperparameters before rendering. They fit the covariance function estimated by casting rays towards the scene from the visible points. These global hyperparameters are then used for all the computations performed during the rendering pass (where incoming irradiance is computed

for each visible point). However, we think that locally determined hyperparameters would provide better results. In this way, we would determine a different set of optimal hyperparameters for each visible point. To this end, we could compute sets of quadrature vectors for a set of fixed values of hyperparameters, then compute the value of the likehood function at runtime for each set. The set yielding the best likehood is then chosen for rendering.

6.2.2.2 Glossy surfaces

Our current implementation of Bayesian Monte Carlo considers only diffuse surfaces. The glossy reflections are computed using classical importance Monte Carlo. BMC could be interesting for computing a fast approximation of glossy reflections with few uniformly distributed rays. Indeed, recall that Bayesian Monte Carlo reconstructs an approximation of the function $f(x)$ (regression), given a set of data. The BMC integration yields the exact integral of the product of the regression function and the function $p(x)$. Consequently, even with a sparse sampling, BMC could provide a smooth (*i.e.* less noisy) approximation of glossy reflections. Of course, Bayesian Monte Carlo used in conjunction with importance sampling would yield better results. However, the extension to glossy surfaces is costly, because it requires to generate additional directions sets and to evaluate a high dimensional function at runtime.

6.2.2.3 Adaptive sampling

Most Monte Carlo algorithms allows adaptive sampling. The decision of taking additional samples is generally based on the variance of the already obtained samples. If the variance is higher than a given threshold, more samples are taken. Usually, the additional computation needed to compute the new estimate from the new sample values is low. This is not the case in Bayesian Monte Carlo, where the computation of a new estimate requires to invert a larger matrix. In addition, it requires to be able to know the direction and the values associated to all the previous samples. However, progressivity could be integrated during the preprocessed step. Instead of precomputing samples sets composed of n samples, we could build a hierarchy of sets. These sets would be built by iteratively adding sample directions, then computing the associated quadrature vector, and so on. At rendering time, the appropriate level of the hierarchy is used.

6.2.2.4 Path tracing

The application of Bayesian Monte Carlo to final gathering only takes into account the directional covariance properties of the incoming radiance. Indeed, the correlation between two sample values depends only on the angular distance between their associated directions. We would like to extend this approach to path-tracing. In this case, we would consider a higher dimension integrand and a larger number of hyperparameters. Compared to our implementation, a simple extension of Bayesian Monte Carlo to path tracing would require a covariance function which depends on one additional parameter, lp_0.

Bibliography

[BDT99] Kavita Bala, Julie Dorsey, and Seth Teller. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Transactions on Graphics*, 18(3):213–256, 1999.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communication on the ACM*, 18(9):509–517, 1975.

[Bir62] Allan Birnbaum. On the foundations of statistical inference. *Journal of the American Statistical Association*, 57(298):269–306, 1962.

[CAE08] David Cline, Daniel Adams, and Parris K. Egbert. Table-driven adaptive importance sampling. *Computer Graphics Forum*, 27(4):1115–1123, 2008.

[Chr99] Per H. Christensen. Faster photon map global illumination. *Journal of Graphics Tools*, 4(3):1–10, 1999.

[DBB02] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[DLAW01] R.O. Dror, T.K. Leung, E.H. Adelson, and A.S. Willsky. Statistics of real-world illumination. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2001.*, volume 2, pages 164–171, 2001.

[FCH+06] Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu chi Lai. Optimizing control variate estimators for rendering. *Computer Graphics Forum*, 25(3):351–357, 2006.

[FD09] B. Fabianowski and J. Dingliana. Interactive global photon mapping. *Computer Graphics Forum*, 28(4):1151–1159, 2009.

[FMH05] David Fradin, Daniel Meneveaux, and Sebastian Horna. Out of core photon-mapping for large buildings. In *Proceedings of Eurographics Symposium on Rendering 2005*. Eurographics, June 2005.

[GBP06] Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Temporal radiance caching. Technical Report 1796, IRISA, Rennes, France, 2006.

[GKBP05] Pascal Gautron, Jaroslav Krivanek, Kadi Bouatouch, and Sumanta Pattanaik. Radiance cache splatting: a gpu-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering*, page 36, 2005.

[Gla94] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, July 1984.

[Hec90] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *SIGGRAPH Computer Graphics*, 24(4):145–154, 1990.

[Her04] Aaron Hertzmann. Introduction to bayesian learning. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, page 22, New York, NY, USA, 2004. ACM.

[HHS05] Vlastimil Havran, Robert Herzog, and Hans-Peter Seidel. Fast final gathering via reverse photon mapping. *Proceedings of Eurographics)*, 24(3):323–333, 2005.

[HLO04] Fred J. Hickernell, Christiane Lemieux, and Art B. Owen. Control variates for quasi-monte carlo. *Statistical Science*, 20, 2004.

[HP02] Heinrich Hey and Werner Purgathofer. Advanced radiance estimation for photon map global illumination. *Computer Graphics Forum (Proceedings of Eurographics 2002)*, 21(3), September 2002.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30. Springer-Verlag/Wien, 1996.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Natick, MA, 2001.

[JMY90] M.E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26:131–148, 1990.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.

[KBPZ06] Jaroslav Krivanek, Kadi Bouatouch, Sumanta N. Pattanaik, and Jiri Zara. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Proceedings of Eurographics Symposium on Rendering*, 2006.

[Kel97] A. Keller. Instant radiosity. In *Proceedings of SIGGRAPH*, pages 49–56, 1997.

[KGBP05] Jaroslav Krivanek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved radiance gradient computation. In *Proceedings of Spring Conference on Computer Graphics*, pages 155–159, 2005.

[KGPB05] Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. Technical Report 5, 2005.

[KNKS08] Apurva Kumar, Prasanth B. Nair, Andy J. Keane, and Shahrokh Shahpar. Robust design using bayesian monte carlo. *International J. for Numerical Methods in Engineering*, 73(11):1497–1517, 2008.

[KW00] Alexander Keller and Ingo Wald. Efficient importance sampling techniques for the photon map. In *Proceedings of Vision, Modeling and Visualization 2000*, pages 271–279, Saarbruecken, Germany, November 2000. IOS Press.

[Lew94] Robert R. Lewis. Making shaders more physically plausible. *Computer Graphics Forum (Eurographics '94 Conference Issue)*, 13(3):1–13, 1994.

[LSK+07] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering*, pages 227–286, 2007.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93*, pages 145–153, 1993.

[LW94] Eric Lafortune and Yves D. Willems. The ambient term as a variance reducing technique for monte carlo ray tracing. In *Proceedings 5th Eurographics Workshop on Rendering*, pages 163–171. Springer, 1994.

[LW95] Eric Lafortune and Yves Willems. A 5D tree to reduce the variance of monte carlo ray tracing. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 11–20. Springer-Verlag, 1995.

[LZT+08] Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François Sillion, and Timo Aila. A meshless hierarchical representation for light transport. In *Proceedings of SIGGRAPH 2008)*, volume 27, 2008.

[MBC00] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[McC94] Ross McCluney. *Introduction to Radiometry and Photometry*. Artech House, Inc., Norwood, MA, USA, 1994.

[Min00] T P Minka. Deriving quadrature rules from gaussian processes. Technical report, Statistics Department, Carnegie Mellon, 2000.

[Mit87] Don P. Mitchell. Generating antialiased images at low sampling densities. In *SIGGRAPH*, pages 65–72, 1987.

[Mit96] Don P. Mitchell. Consequences of stratified sampling in graphics. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 277–280, New York, NY, USA, 1996. ACM.

[Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, 1992.

[O'H87] A. O'Hagan. Monte-carlo is fundamentally unsound. *The Statistician*, 36(2/3):247–249, 1987.

[O'H91] A. O'Hagan. Bayes-hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

[OZ00] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, March 2000.

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering*. Morgan Kaufmann, 2004.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[PHR06] T. Pfingsten, D. J. L. Herrmann, and C. E. Rasmussen. Model-based design analysis and yield optimization. *Semiconductor Manufacturing, IEEE Trans. on*, 19(4):475–486, 2006.

[PS89] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. *SIGGRAPH Computer Graphics*, 23(3):281–288, 1989.

[PTVF88] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - THe Art of Scientific Computing*. Cambridge University Press, 1988.

[RG02] Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. In *Neural Information Processing Systems*, pages 489–496. MIT Press, 2002.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Process for Machine Learning*. MIT Press, 2006.

[Sch03] Roland Schregle. Bias compensation for photon maps. *Computer Graphics Forum*, 22(4), 2003.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, April 1986.

[Sim98] Jeffrey S. Simonoff. *Smoothing Methods in Statistics (Springer Series in Statistics)*. Springer, May 1998.

[SK97] E.B. Saff and A.B.J. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, 1997.

[Sob93] I. M. Sobol. Sensitivity analysis for non-linear mathematical models. *Mathematical Modeling & Computational Experiment*, 1(4):407–414, 1993.

[SSS01] Annette Scheel, Marc Stamminger, and Hans-Peter Seidel. Thrifty final gather for radiosity. In *Rendering Techniques 2001 (Proceedings of Eurographics Workshop on Rendering 2001)*. Eurographics, Spinger-Verlag, June 2001. held in London, UK.

[SSS02] Annette Scheel, Marc Stamminger, and Hans-Peter Seidel. Grid based final gather for radiosity on complex clustered scenes. In *Computer Graphics Forum, Proceedings of Eurographics 2002*, volume 21. Blackwell, September 2002.

[SW00] Frank Suykens and Yves D. Willems. Density control for photon maps. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 23–34. Springer Wien, 2000.

[TJ97] Rasmus Tamstorf and Henrik Wann Jensen. Adaptive sampling and bias estimation in path tracing. In *In Eurographics Rendering Workshop*, pages 285–295, 1997.

[Vea98] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. Adviser-Guibas, Leonidas J.

[VG95] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on computer graphics and interactive techniques*, pages 419–428, 1995.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. *Computer Graphics*, 31(Annual Conference Series):65–76, 1997.

[War94] Gregory J. Ward. The RADIANCE Lighting Simulation and Rendering System. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 459–472, 1994.

[WGS04] Ingo Wald, Johannes Gunter, and Phillip Slusallek. Balancing considered harmful - faster photon mapping using the voxel volume heuristic. *Computer Graphics Forum (Proceedings of Eurographics 2004)*, 23(3), 2004.

[WH92] Greg Ward and Paul Heckbert. Irradiance gradients. In *In Proceedings of Eurographics Workshop on Rendering*, pages 85–98, 1992.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 85–92, August 1988.

List of Figures